E.T.S. de Ingeniería Industrial, Informática y de Telecomunicación

DESARROLLO DE UNA BASE DE DATOS BASADA EN LA ESTIMACIÓN DE LA DIRECCIÓN DE LA MIRADA Y LA POSICIÓN DE LA CABEZA

Grado en Ingeniería
en Tecnologías de Telecomunicación

Trabajo Fin de Grado

Saray Agorreta Cervantes

Rafael Cabeza Laguna

Sonia Porta Cuéllar

Pamplona, 20/06/2016

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

DESARROLLO DE UNA BASE DE DATOS BASADA EN LA ESTIMACIÓN DE LA DIRECCIÓN DE LA MIRADA Y LA POSICIÓN DE LA CABEZA

AUTORA: Saray Agorreta Cervantes
TUTORES: Rafael Cabeza Laguna
Sonia Porta Cuéllar

Grupo de Investigación de Ingeniería Biomédica

Dpto Ingeniería Eléctrica y Electrónica

UPNA

Junio de 2016

INDICE

RESUMEN

Para la elaboración de la BBDD se parte del algoritmo existente basado en la estimación de la dirección de la mirada y de la posición de la cabeza. Resulta necesario revisarlo y modificarlo: flexibilizarlo pudiendo elegir los índices de los puntos a mirar, revisar almacenamiento y formato de los datos de salida, guardar las coordenadas de los puntos mirados y optimizar la parte de adquisición de datos e imágenes de manera que sea más eficiente la posterior grabación.

Finalmente se realizará la grabación de 2 usuarios que conformarán la BBDD de prueba en diferentes disposiciones: posición central moviendo libremente la cabeza, posición central moviendo la cabeza de forma estática y con traslaciones derecha e izquierda y delante y atrás moviendo la cabeza libremente.

PALABRAS CLAVE

Trakstar, sensor, cámara, imágenes, mirada, cabeza, posición, optimización, sistema de referencia, sincronización, plantilla, tiempo, base de datos

ABSTRACT

The purpose of this Project is to develop a database using the existing algorithm based on head pose estimation (HPE) and point of regard (POS).

It is necessary to revise and modify the algorithm mentioned: optimize it and make it more flexible doing some changes like increasing the number of points available to look, the storage format and improve the algorithm of sensor data acquisition.

Finally, this test database is formed by two users moving their heads in different positions statically and freely.

KEY WORDS

Trakstar, sensor, camera, images, head, pose, position, optimization, reference system, synchronization, database

INTRODUCCION

La posibilidad de interactuar con una interfaz que disponga de una cámara web sin necesidad de hacer uso de las interfaces típicas destinadas a tal efecto (teclado, ratón, micrófono, etc.) es una atractiva alternativa para situaciones en las cuales es imposible o incómodo utilizar las manos o la voz. Esta situación se puede presentar en el caso de personas con algún tipo de discapacidad física o también en personas que por necesidad o simplicidad prefieran no tener que interactuar de esa forma.

El desarrollo de sistemas de seguimiento de la mirada de bajo coste es uno de los temas de investigación del grupo de ingeniería biomédica en la UPNA.

En este trabajo se revisa y modifica el algoritmo existente de estimación de la posición de la cabeza con el fin de crear una base de datos.

1. ESTADO DEL ARTE

Durante la última década, el campo de la detección de la posición de la cabeza (head pose estimation o HPE) ha sido muy investigado. Consiste en obtener, a partir de una imagen de la cara de una persona, la posición y orientación de la cabeza con respecto a la cámara. A grosso modo pueden clasificarse según su rango de salida en dos grupos [2]:

A) "Coarse head pose estimation systems"

Para estos sistemas la salida está dentro de un rango discreto de las posiciones llamado poses. La pose de salida es una de un conjunto finito de poses previamente definidas. Inicialmente muchos sistemas de estimación de este tipo fueron diseñados, por ejemplo, eran capaces de distinguir dos interlocutores en una conversación.

Estos sistemas no son adecuados para la estimación de la mirada, ya que no son capaces de representar el movimiento continuo inherente a la naturaleza de esta.

B) "Fine head pose estimation systems"

En este caso, la salida de estimación de la posición es continua, es decir, la pose estimada es analógica.

A este grupo pertenecen la mayoría de las soluciones que dan una mejor precisión, y son muy adecuados como paso previo para la estimación de la mirada.

Desde un punto de vista más técnico, Murphy-Chutorian y Trived [1] clasifican la estimación de la posición de la cabeza en 8 categorias:

I) "Appearance template methods"

Este método compara una nueva imagen de la cabeza con un conjunto de ejemplares con diferentes poses con el fin de encontrar la más similar. El uso de este método resulta ventajoso debido a su gran simplicidad y a la capacidad de procesar imágenes de baja resolución.

II) "Detector array methods"

Este método es similar al anterior, pero en vez de utilizar la imagen con todos los casos de poses predefinidas para encontrar la más probable, se utiliza un clasificador binario previamente preparado para las caras que comparten la misma pose.

III) “Nonlinear regression methods”

Se utilizan herramientas de regresión no lineal para desarrollar un mapa funcional de la imagen o información relacionada con la medida de la posición de la cabeza.

IV) “Manifold embedding methods”

Este método trata de modelar la variación continua del movimiento de la cabeza en unas pocas dimensiones teniendo en cuenta la cabeza como un modelo rígido (3 para orientación y tres para la posición).

V) “Flexible models”

Esta técnica se basa en el ajuste de un modelo flexible a la estructura de la cara en el plano imagen. La forma de esta se utiliza para estimar su posición.

VI) “Geometric methods”

Los métodos geométricos utilizan la ubicación de los ojos, boca y punta de la nariz para determinar la pose.

VII) “Tracking methods”

Esta técnica estima la posición de la cabeza obteniendo la información con respecto a dos frames.

VIII) “Hybrid methods”

Los métodos híbridos combinan uno o más de los métodos comentados anteriormente para superar limitaciones inherentes propias de un método individual.

2. RECURSOS EMPLEADOS

Para la elaboración de este trabajo se ha contado con los siguientes recursos.

2.1. HUMANOS

Usuarios para la grabación de la base de datos.

2.2. MATERIAL EMPLEADO

A continuación, se explica brevemente el material utilizado,

2.1.1. 3D Guidance trakSTAR (Ascension Technology Corporation)

Este dispositivo está formado por tres elementos principales:

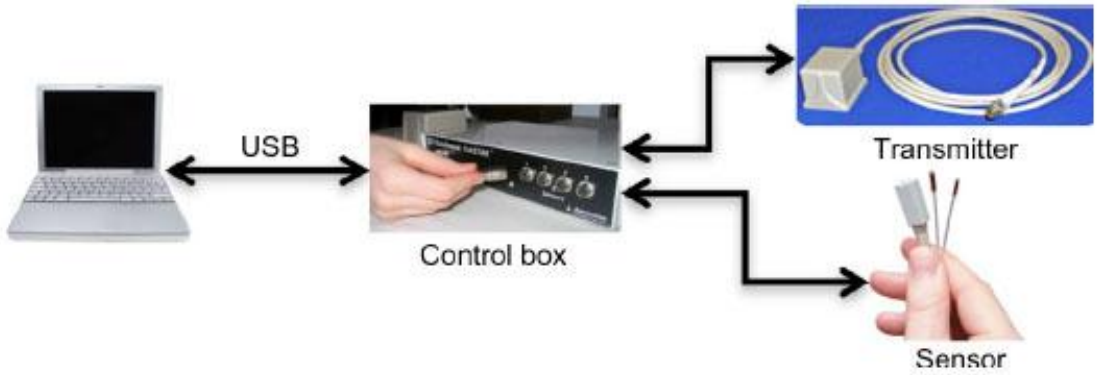

Fig. 1 - Esquema TrakStar

Transmisor: Emite campos electromagnéticos que los sensores detectan.

Unidad electrónica: Se encarga de procesar la información de los sensores, controlar y alimentar el transmisor y comunicarse con el ordenador.

Sensores: Perciben la señal emitida por el transmisor y estiman su translación y rotación con respecto al sistema de coordenadas del transmisor enviándolo a la unidad electrónica.

La unidad principal genera un campo electromagnético permitiendo saber la posición de los sensores conectados: rotación y translación con respecto al sistema de coordenadas del transmisor. Esta información se envía al ordenador mediante una conexión USB.

2.2.2 Piezas para el soporte de los sensores

Para el manejo de los sensores de manera precisa se dispone de 2 piezas de plástico rígido.

La primera de ellas necesaria para marcar los puntos en el espacio tanto en la calibración de la plantilla como para el marcado de los puntos faciales.

Fig. 2.1 - Pieza útil de marcado

Fig. 2.2 - Pieza cabeza

La segunda pieza se utiliza durante las grabaciones para colocar el sensor en la cabeza del usuario. Se acopla a una diadema de manera firme, de modo que durante todo el proceso de grabación el sensor permanece solidario con la cabeza.

2.2.3. Cámara web Logitech HD Pro C920

Fig.3 - Cámara web

Permite grabaciones de 1280x720 a una frecuencia de adquisición de 30 frames por segundo. La resolución real máxima es de 1920x1080.
La cámara se utiliza en el proceso de calibración del sistema, así como en la calibración del usuario y en el proceso de adquisición de las imágenes.

2.2.4. Modelos de cabeza

Usado para probar el algoritmo en diferentes posiciones.

Fig.4 - Modelo cabeza

2.2.5. Damero

Se utiliza sobre un trípode para la calibración de la cámara y cámara-sensor, con un escaque de 30mm.

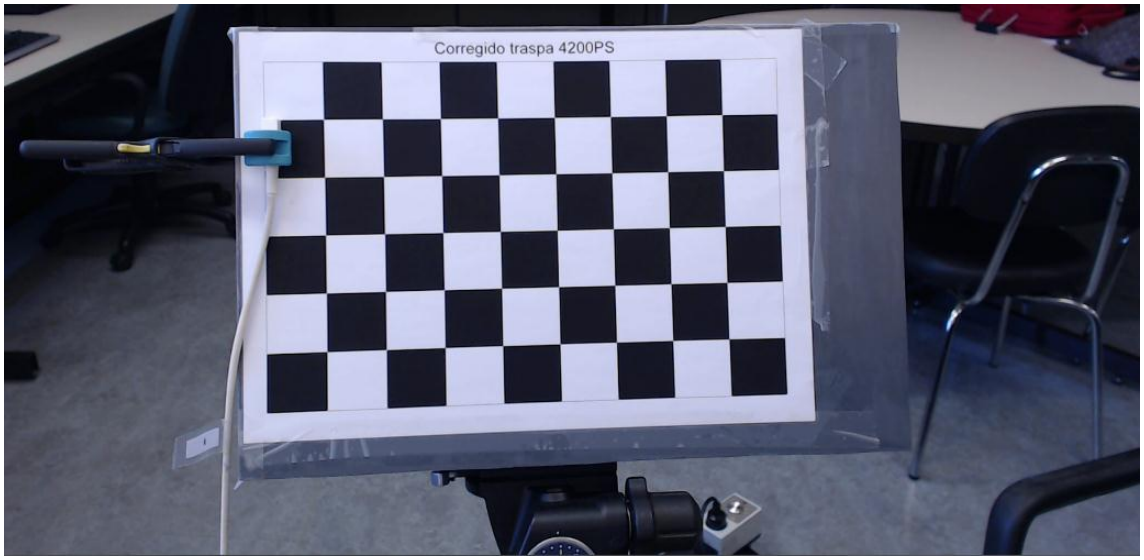

Fig.5 – Damero utilizado durante la calibración del sistema

2.2.6. Modelo de plantilla y soporte de madera

Se ha diseñado un soporte de madera para fijar la plantilla de manera que permanezca fija durante toda la sesión y a una altura adecuada para el usuario.

La plantilla se utiliza a modo de pantalla donde el usuario mira los diferentes índices. En ella se disponen 65 puntos de diferentes colores.

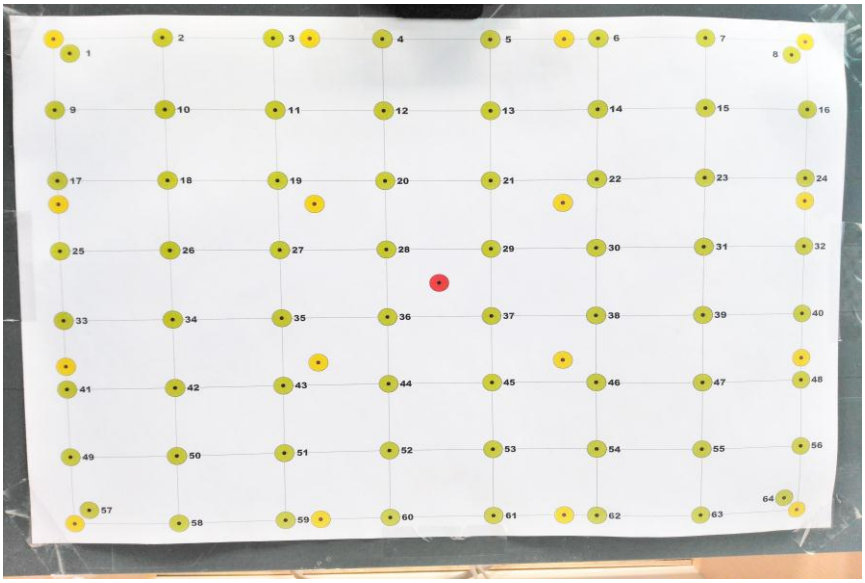

Fig.6 – Plantilla utilizada

2.2.7. Sistema de iluminación

Las diferentes sesiones se han realizado en condiciones de iluminación controladas. Para ello se ha prescindido de la iluminación natural y se ha utilizado únicamente iluminación de estudio.

Se han utilizado 3 focos con sus correspondientes *dimmers* (reguladores de intensidad), un reflector y un difusor, con el objetivo de generar luz difusa sin sombras y una buena iluminación ambiental para dar un mayor realismo a las imágenes.

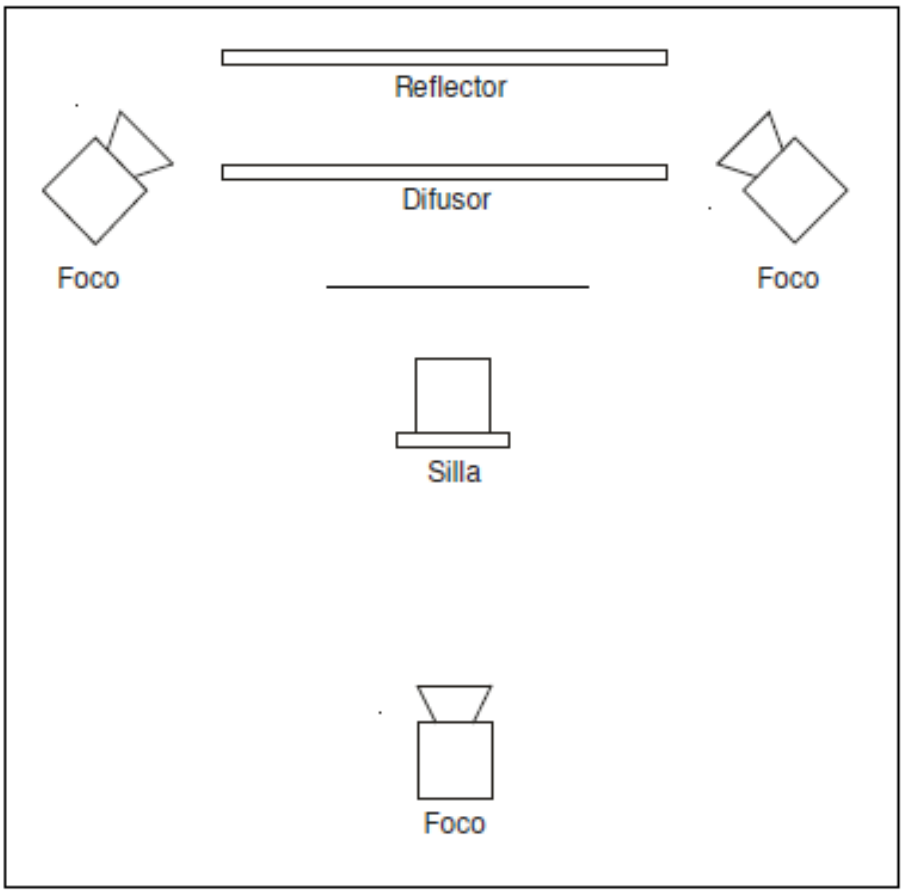

Fig.7 - Esquema iluminación controlada

3. ALGORITMO INICIAL

En este apartado se detalla el funcionamiento del algoritmo de partida [4], para después explicar las modificaciones realizadas.

El algoritmo original está implementado en Matlab2013 y Windows XP.

Todas las funciones están localizadas en la carpeta **"2_Database_____Functions_2013b"**

3.1.- Estructura del algoritmo

A continuación, se muestra el menú principal el cual se corresponde con la función "a0__final_program_3D_iris.m". Una vez ejecutada se muestra lo siguiente.

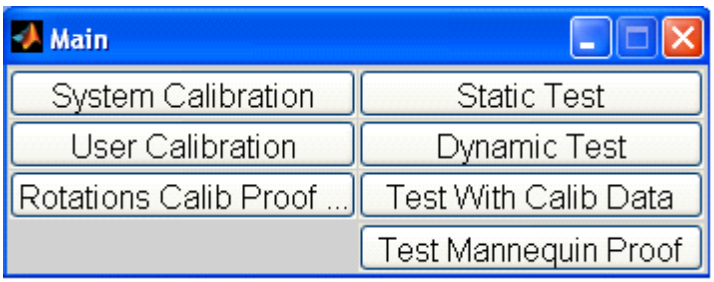

Fig. 8 – Menú principal

A través de este menú se accede a todas las funcionalidades para el desarrollo del algoritmo, las cuales serán explicadas a continuación siguiendo su orden de uso.

3.1.1. System Calibration

Esta opción es la primera a realizar y nos dirige a calibración del sistema: plantilla y cámara-sensor. Para que todo el sistema funcione resulta imprescindible llevar a cabo una buena calibración. Se corresponde con la función "**a1__calibration_y__optimization_damero_sin_marcarlo_y_de_la_plantilla.m**". Esta función se compone a su vez de 5 subfunciones:

```
%---------------------------------------%
%- CALIBRATION FUNCTION + OPTIMIZATION -%
%---------------------------------------%

% Clean this mess
clear, clc, close all;

% Aquí obtenemos los datos para de calibración
calibration__capture_data_v2_online;

% Obtenemos los parámetros de la cámara.
calibration__camera_parameters;

% Calculamos y optimizamos la relación sensor a cámara.
% calibration__dat_2_mat;
calibration_optimization__camera_transmitter_optimizer; % optim_init
%% Hasta aquí es código de Jose Javier

% Marcamos los 81 puntos de la plantilla
calibration_optimization__capture_grid_points;

% Realizamos una corrección de la posición de la plantilla en SC de la
% cámara.
calibration_optimization__grid_alignment_correction;
```

Fig. 9 – Subfunciones para la calibración del sistema

Las tres primeras son para la calibración del sistema cámara-sensor.

Previamente a la ejecución de la primera función, ***"calibration__capture_data_v2_online.m"***, se coloca el sensor en la posición indicada para ello del damero (primera casilla negra superior izquierda), y éste a su vez en el soporte del trípode del laboratorio. Es importante e imprescindible la utilización de la **pinza** para asegurarnos que el sensor permanece fijo durante todo el proceso de calibración. Por otro lado, también resulta imprescindible activar la opción de autoenfoque desde la propia aplicación de la cámara para obtener unos resultados óptimos.

Teniendo el soporte preparado debemos tomar 50 imágenes de 50 posiciones distintas del damero en el espacio durante el transcurso de esta función presionando cualquier tecla para la toma de la imagen correspondiente. Estas imágenes se guardan en la carpeta ***"2_Database_____Data__Calibration"***.

Las siguientes dos funciones simplemente deben ejecutarse para que se optimicen los datos de calibración y se extraigan los parámetros intrínsecos de la cámara tales como la distancia focal, el punto principal, los coeficientes de inclinación que definen el ángulo entre los pixeles de los ejes x e y, y sus distorsiones [3]. Estos parámetros se guardan cada uno en un archivo .mat.

Posteriormente se almacenarán en una estructura (**apartado 4.1 a**).

La cuarta función, ***"calibration_optimization__capture_grid_points.m"***, permite referenciar el sistema de coordenadas de la plantilla en función del de la cámara. Para ello al ejecutar la función, se mostrará un diálogo en un terminal independiente de matlab que nos indicará que debemos pulsar "enter" cuando estemos tocando con la punta del "útil de marcado" el punto que queramos representar en el sistema de coordenadas de la cámara.

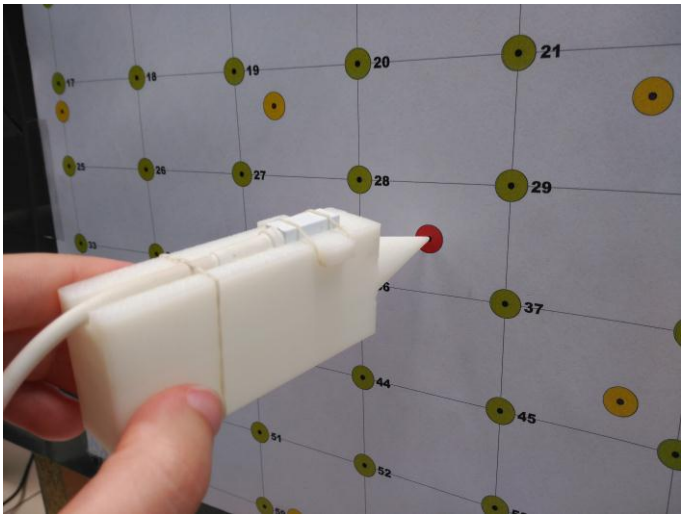

Fig. 10 – Marcado plantilla

Para este marcado se debe seguir el siguiente orden: en primer lugar, los 64 puntos verdes, después los 16 amarillos y para terminar el punto rojo central. Testeando el algoritmo se representan los puntos reales y los estimados en el marcado de la plantilla.

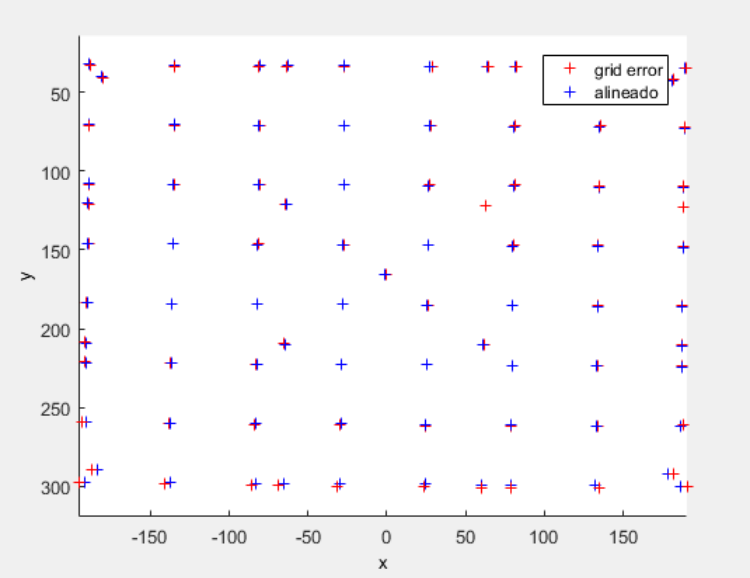

Fig. 11 – Error marcado de la plantilla

Se aprecia en la parte inferior un error considerable el cual es debido a la excesiva distancia que hay entre el sensor colocado en esa fila hasta el transmisor. Esto limita la exactitud de los resultados finales, se soluciona en el **apartado 4.2**.

Una vez terminado el sistema se cerrará y deberemos ejecutar la última función finalizando con ella la calibración del sistema:

"**calibration_optimization__grid_alignment_correction.m**" nos permite optimizar el marcado utilizando el archivo **"Rejilla_real.mat"** (almacenado en la carpeta "**2_Database_____Data__Calibration**") que contiene la disposición en el plano de la plantilla de los 81 puntos en forma matricial (81x3, siendo la tercera componente 0) y estando el origen en el punto superior derecho amarillo (índice 68).

3.1.2. User calibration

Esta opción se corresponde con la función ***"a2___recording_corneas.m"*** a través de la cual accedemos al siguiente menú gráfico:

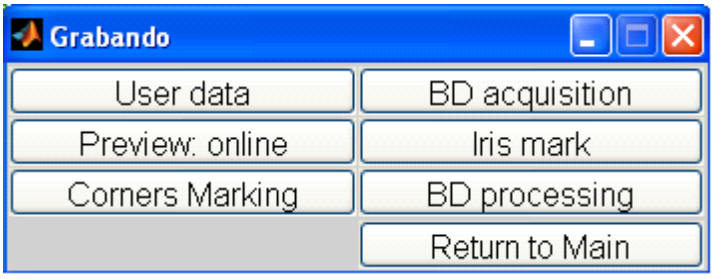

Fig. 12 – User Calibration

3.1.2.1. User data

Se corresponde con la función *"**recording__user_data.m**"*.

Esta función permite llevar un registro de los distintos usuarios. En el caso de ser un nuevo usuario tendrá que introducir una serie de datos: identificador, nombre, apellido, fecha de nacimiento y ciudad de nacimiento. En caso de que ese usuario exista se nos advertirá de que los datos se sobrescribirán. Se guardará la variable N_user y se creará una carpeta por nuevo usuario: ***"2_Database_____Data__Recording/user_'N_user' "***. Dichos datos se almacenarán en un archivo de texto en la carpeta, posteriormente esto se modificará almacenándolos en forma de estructura (**apartado 4.1 b**).

Una vez creada la carpeta se copia la información obtenida en la calibración: las matrices de transformación entre el sistema de coordenadas de la cámara y el sensor, los parámetros intrínsecos de la cámara y la posición de la punta del "útil de marcado" respecto al sensor.

3.1.2.2. Corners Marking

Se corresponde con la función "**recording__capture_corners_points(N_user).m**".

Se introduce N_user como variable de entrada para situarnos en el directorio adecuado.

Previamente a la ejecución de la función se debe colocar el sensor 1 en la diadema sobre la cabeza del usuario y mediante el "útil de marcado" se llevará a cabo el marcado de los puntos de la cara del usuario que sean de interés. El programa nos irá indicando que presionemos enter cuando estemos marcando un punto. Para este caso se marcan 5 puntos: primero los córners de los ojos de izquierda a derecha (izquierda del usuario) y por último la punta de la nariz.

Una vez finalizada la función los datos se guardarán en la carpeta correspondiente con el siguiente nombre: "**user_'N_user '_face_s1.mat**".

3.1.2.3. Preview: Online

Esta opción se corresponde con la función "**recording__preview_online(N_user).m**".

Permite ver en la pantalla los datos adquiridos por el sensor superpuestos a la imagen captada por la cámara web a modo de "comprobación". Se representan los puntos marcados anteriormente y los ejes de coordenadas en el origen del sensor, así como los datos numéricos correspondientes a la posición de la cabeza: coordenadas con respecto a la cámara (X,Y,Z) y ángulos (Roll, Yaw, Pitch).

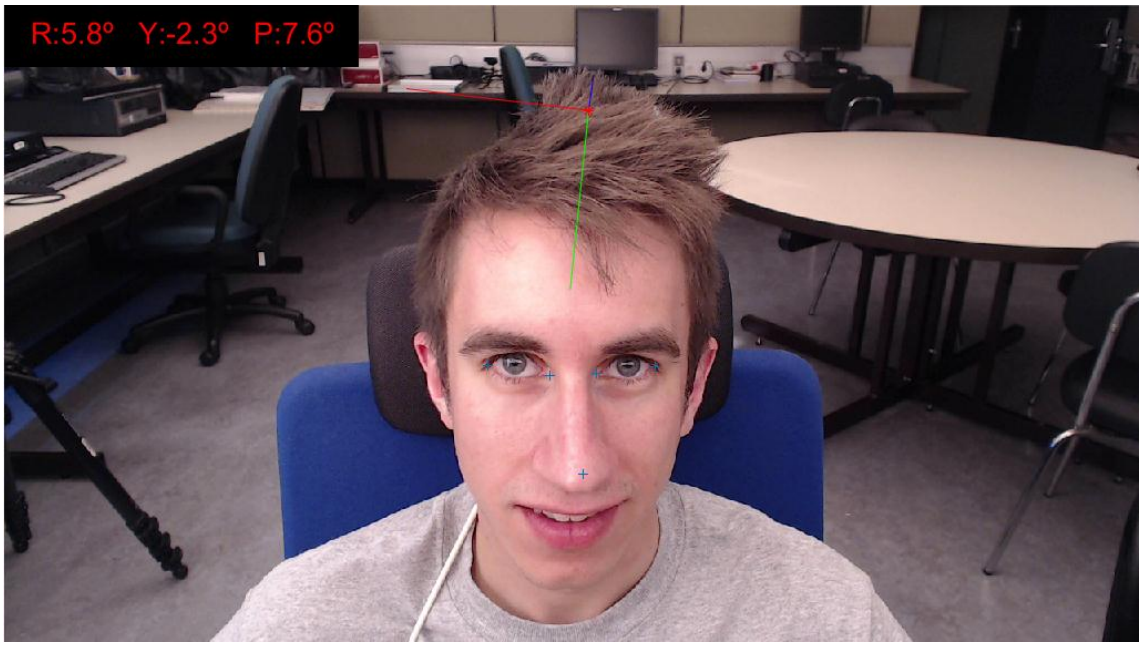

Fig. 13 – Preview online de un usuario

3.1.2.4. BD Adquisition

Esta función se corresponde con *"recording__data_aquisition2_prueba(N_user).m"*.

Durante esta función se realiza una doble adquisición de imágenes y de datos del sensor. Son 30 imágenes por cada una de las 16 posiciones a las que el usuario mira. Estas posiciones deben ser los puntos **amarillos** de la plantilla. Esta limitación en cuanto al número de índices a mirar se resuelve en el (**apartado 4.3**). A su vez, resulta interesante almacenar las coordenadas de los índices mirados referentes a la plantilla para posteriores comprobaciones (**apartado 4.4**).

Se deberá hacer de la siguiente manera por cada punto: Presionar "enter" con el terminal del sistema seleccionado (que corresponde al programa en C++ controlador del TrackStar) y acto seguido presionar "enter" en la ventana de Matlab, produciéndose de esta manera una doble adquisición de datos e imágenes por cada punto. Es importante comprobar que se esté haciendo referencia a la misma posición para poder sincronizar los datos correctamente.

Se propone la modificación en la forma de adquisición (**apartado 4.5.c**) ya que en la práctica puede dar lugar a equívoco perdiendo la sincronía entre la captura de imágenes de un punto y su correspondiente información de posición. Esta pérdida hace inválida la sesión, por tanto, será necesario eliminarla y repetirla de nuevo.

Para cada punto, esperar a que el sistema avise mediante un aviso sonoro de que podemos pasar a la siguiente posición y volver a repetir el proceso para las 16 posiciones.

El transmisor está configurado para que recoja 3600 muestras por cada posición a una frecuencia de 240 Hz. De esta manera obtenemos para el trakStar un tiempo de procesado por punto de 15 segundos y para el procesado de imágenes un tiempo de 17.76 segundos. Resulta extremadamente lento, por lo que es necesario revisar el guardado de las imágenes (**apartado 4.5.a**) y inicialización de la cámara (**apartado 4.5.b**) para que sea más eficiente.

Para el correcto funcionamiento esta función necesita una serie de datos generados durante la calibración. Estos datos son leídos directamente de la carpeta correspondiente.

A la finalización de esta función tendremos una serie de datos que se localizarán dentro de la carpeta de usuario (**2_Database_____Data__Recording/user_'N_user'**), en la subcarpeta correspondiente a la posición concreta (**/posicion_'N_posicion'**), que va del 1 al 16, y a su vez en la subcarpeta correspondiente al número de captura (**/imagen_'N_captura'**), que va del 1 al 30.

Los datos que se guardan en cada carpeta son los siguientes:

1) El HPE del sensor de la diadema: en forma de vector de dimensión 6. Los tres primeros datos corresponden con la traslación en X, Y y Z respecto de la cámara. Los otros 3 son los datos de rotación **del sistema de coordenadas del sensor al de la cámara**. Se guardan en el siguiente archivo:

"**user_'N_user'_posicion_'N_posicion'_HPE_sensor.mat**".

2) Traslación HPE: Traslación del sensor, vector fila de dimensión 3. Se guarda en el siguiente archivo:

"**user_'N_user'_posicion_'N_posicion'_traslaciónHPE.mat**".

3) Rotación HPE: Datos de rotación del sensor pero en este caso **del sistema de coordenadas del transmisor al de la cámara**. Vector columna de dimensión 3. Se guarda en el siguiente archivo:

"**user_'N_user'_posicion_'N_posicion'_rotacionHPE.mat".**

4) Coordenadas de los puntos de la cara en el plano imagen: Se guardan las posiciones de las proyecciones de los puntos marcados de la cara en el plano imagen. Se guardan en formato de matriz 5x2 (5 filas de 2 columnas con las

posiciones X e Y de cada uno de los 5 puntos de la cara marcados, el orden es de arriba a abajo y se corresponden con los "corners" de los ojos de izquierda a derecha y en último lugar la punta de la nariz. Se guardan en el siguiente archivo:
"**user_'N_user'_posicion_'N_position'_face_image_sensor.mat**".

4. MODIFICACIONES

A continuación, se exponen los cambios realizados con respecto al algoritmo inicial.

4.1 Almacenamiento y formato de salida

a) Parámetros intrínsecos de la cámara en estructura

Se ha revisado el almacenamiento de los parámetros de la cámara de manera que en lugar de guardarlos en variables separadas se guardan en una estructura con el nombre CameraParams siendo cada parámetro (focal_length, principal_point, skew, etc) un campo diferente de esta.

b) Datos usuario en estructura

Se modifica la forma de guardar los datos del usuario de
manera que en lugar de guardarlos en un archivo de texto se guardan en una estructura (user) con los campos name, surname, date y place.

4.2 Comprobación altura pantalla con respecto al transmisor

La solución adoptada ha sido construir una estructura de madera de manera que se pueda modificar la altura de la plantilla permitiendo acercarla al transmisor aumentando a su vez la estabilidad y sujeción de esta. Dicha plantilla hace el papel de futura "pantalla" a la que el usuario mirará.

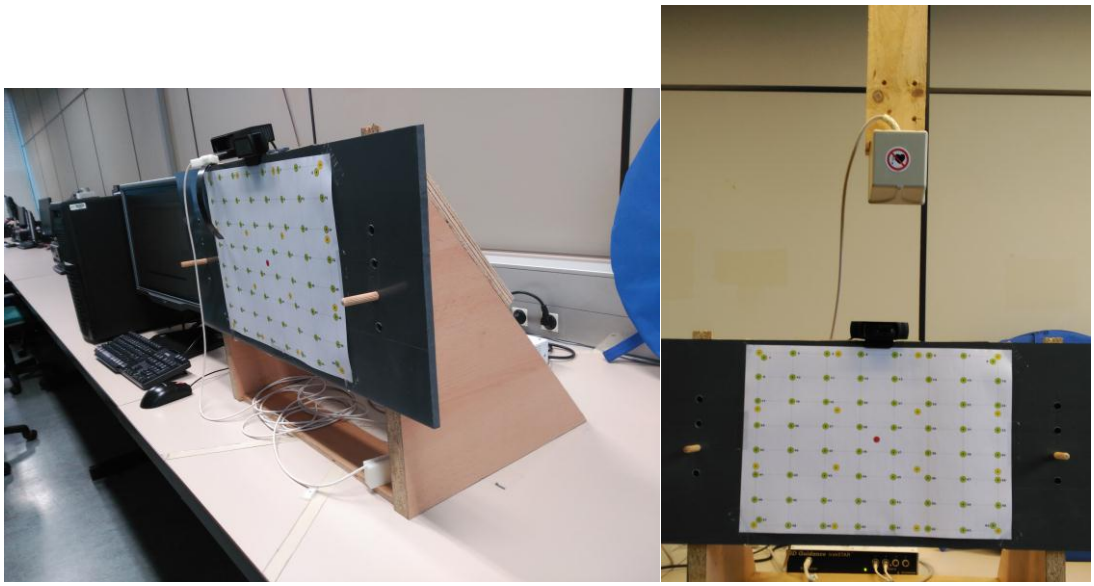

Fig. 14 – Disposición de la "pantalla"

A continuación, observamos la corrección en cuanto al error de marcado en la fila inferior de la plantilla con respecto a la original (comparar con la fig. 11).

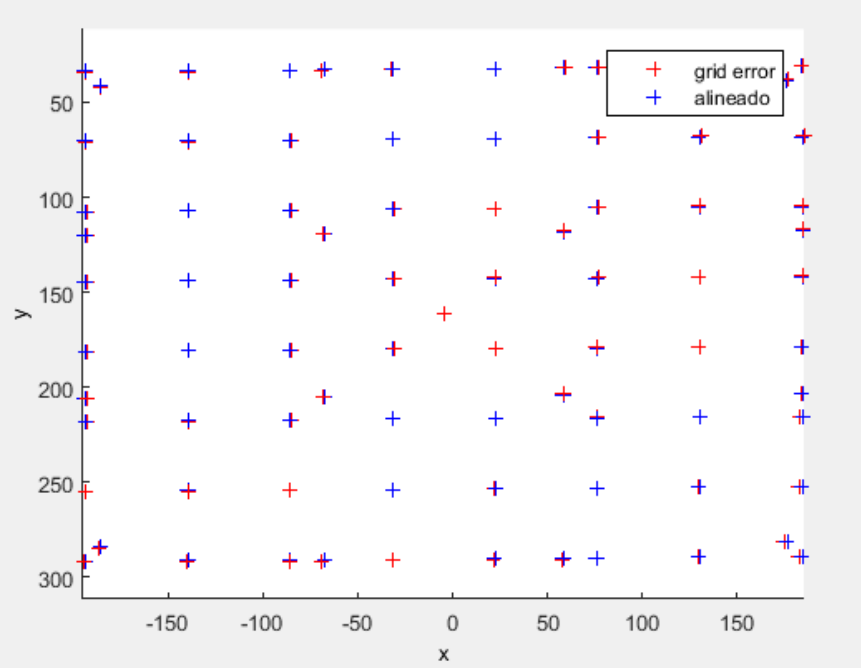

Fig. 15 – Error marcado con respecto a los puntos reales

4.3 Modificación número de posiciones de la cabeza (número de puntos a mirar)

Se implementa una nueva interfaz gráfica la cual nos permite elegir entre las siguientes opciones predeterminadas de acuerdo a la disposición de los puntos en nuestra plantilla.

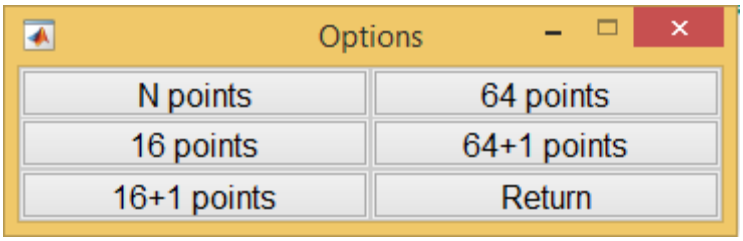

Fig. 16 – Menú BD Adquisition

- N points: se le pide al usuario que introduzca los índices de los puntos a mirar. Pudiendo ser estos cualquiera de los 81 índices disponibles en la plantilla.
- 16 points: referido a los puntos amarillos.
- 16 + 1 points: hacen referencia a los puntos amarillos incluido el punto rojo central.
- 64 points: referido a los 64 puntos verdes.
- 64 + 1 points: referido a los puntos verdes incluido el punto rojo central.

4.4 Almacenamiento de las coordenadas de los índices mirados

Para cada una de las funciones de adquisición comentadas en el punto anterior, se almacena una matriz con el nombre "***coordenadas.mat"*** en la carpeta del usuario correspondiente. De la cual la primera columna muestra los índices mirados y en las siguientes dos columnas las correspondientes coordenadas reales de cada uno referenciadas a la plantilla.

4.5 Optimización del tiempo de adquisición

a) Posponer el guardado de imágenes a png

Tal y como se ha expuesto con anterioridad, el tiempo de adquisición de imágenes resulta más lento que el del almacenamiento de muestras del sensor por lo que se plantean diferentes soluciones hasta llegar a la definitiva:

- Paralelizar offline después de la adquisición: esta opción resulta inviable ya que no se dispone de memoria suficiente para una sesión de 64 puntos (30 imágenes x 64 puntos).
- Paralelizar la escritura a disco de las imágenes entre cada punto. De esta manera se logra un tiempo de adquisición de imágenes para un punto de 7 segundos.
- Paralelizar la escritura a disco de imágenes entre cada punto a mirar guardando la matriz correspondiente a las imágenes de cada punto. De esta forma se consigue ahorrar el tiempo de compresión de imagen de la conversión a png. Finalmente se hace dicha conversión de manera offline y paralelamente también para agilizar el proceso. En este caso obtenemos un tiempo de adquisición de imágenes por punto de unos 6 segundos, siendo esta la solución más eficiente.

b) Revisar inicialización de la cámara

En el código existente se inicializa la cámara para cada índice a mirar. Esto resulta totalmente ineficiente de manera que se inicializa única y exclusivamente antes de comenzar la toma de imágenes. De esta manera conseguimos reducir considerablemente el tiempo de adquisición de imágenes a 1 segundo aprox. por cada punto.

c) Adquisición simultánea de imágenes y datos del sensor.

En este punto el objetivo se centra en disminuir el tiempo de adquisición del trakStar, para lo cual se propone, teniendo como referencia el código de la aplicación original del Trackstar "***run_traker.m"***, implementar una nueva función en Matlab de manera que se le llame paralelamente a la ejecución del programa todo desde la misma ventana de Matlab.

Al inicio de esta nueva función ***"get_points_trak_star.m"*** se hace el reinicio del TrakStar y acto seguido se mantiene recogiendo muestras paralelamente a la adquisición de las imágenes por punto de manera que dejará de coger muestras cuando compruebe la existencia del archivo pdata.mat, matriz correspondiente a las imágenes del último punto de la sesión. Dicha comprobación se hará cada cierto tiempo. De esta manera en la adquisición se limita el tiempo al proceso de captura de imágenes (1 segundo/punto).

De esta manera se comprueba que el sensor no mantiene la frecuencia de muestreo constante, perdiendo muestras, lo que implica una mala sincronización posterior con las imágenes.

Para averiguar el origen de esta pérdida e intentar minimizarla se realizan varias pruebas modificando ciertos parámetros

- Número de workers en la paralelización disponiendo de un máximo de 12.
- Opción número de workers/job disponible en las preferencias de la paralelización.
- Intervalo de tiempo para guardar los datos del sensor
- Comprobar que la CPU no satura en la paralelización del guardado de las imágenes: Se comprueba que cuando se termina la adquisición del punto n+1 se termina de grabar a disco paralelamente la matriz de imágenes correspondientes al punto n, por tanto la CPU se mantiene estable.

Se observa que ejecutando el programa varias veces manteniendo una configuración determinada de estos parámetros la frecuencia de muestreo obtenida varía sin patrón aparente, siendo completamente aleatoria. Ante esta situación para intentar obtener una distribución definida de pérdida de muestras se modifica el código para que se ejecute 20 veces seguidas para la adquisición de 2 puntos y se representa el error en sus correspondientes histogramas. Durante la ejecución surge el problema de que el transmisor al reiniciarse no se conecta correctamente por lo que se repiten las ejecuciones hasta conseguir tener 20 para poder comparar en igualdad de condiciones.

Para cada configuración se calcula la diferencia de tiempos entre muestras consecutivas para cada ejecución obteniendo una matriz de periodos de muestreo (Nºmuestras-1) x 20 ejecuciones. Estando el transmisor configurado a 240Hz este tiempo debiera ser 1/240Hz ≈ 0.0041 segundos.

A continuación, se muestran las diferentes configuraciones testeadas: variando los parámetros anteriormente comentados.

c1) Almacenando los datos del sensor cada 2500 muestras recogidas.

- **3 workers + 1 worker/job**

Periodo de muestreo máximo (s)	0.4927
Periodo de muestreo mínimo (s)	0.0017
0.0043 < Nº valores < máximo	737 /4999
Tiempos de muestreo	0,0017 0,0029 0,0033 0,0037 0,0040 0,0041 0,0042 0,0065 0,0066 0,0082 0,0083 0,0084 0,0165 0,0167 0,0249 0,0250 0,0836 0,4797

Tabla 1 – 3 workers [1 1]

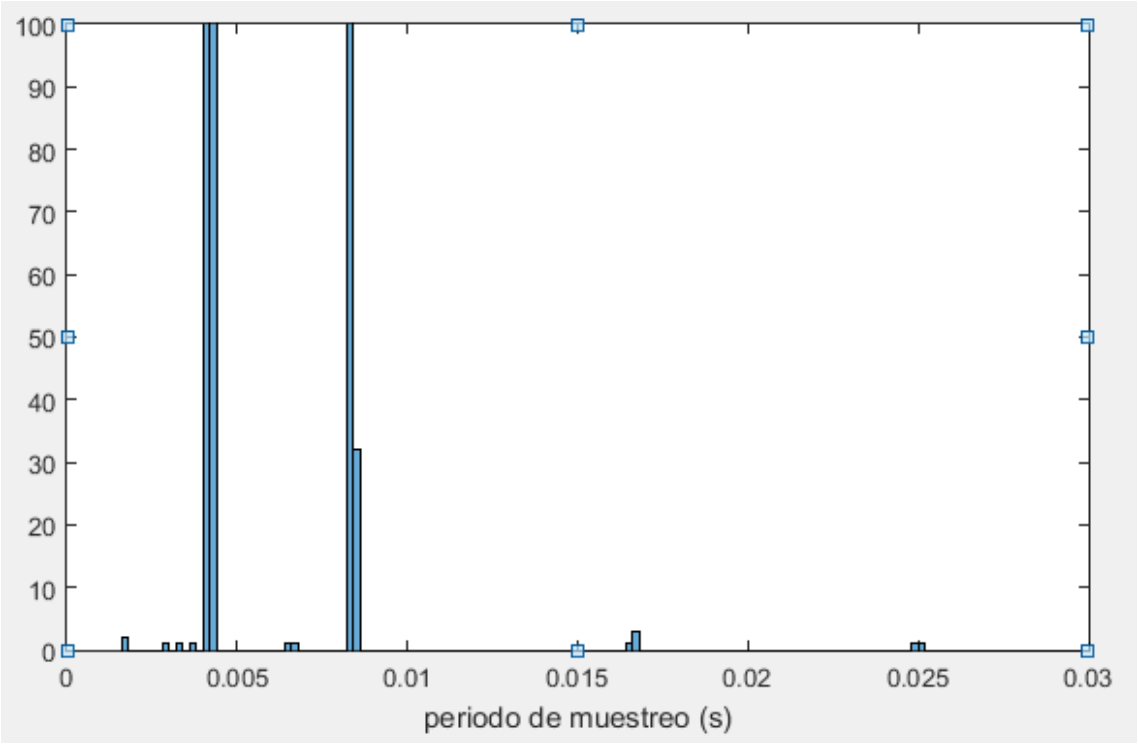

Fig. 17 – Histograma 3 workers [1 1]

- **12 workers**

Periodo de muestreo máximo (s)	18.0547
Periodo de muestreo mínimo (s)	0.0025
0.0043 < Nº valores < máximo	442 /4999
Tiempos de muestreo	0,0025 0,0040 0,0041 0,0041 0,0042 0,0082 0,0083 0,0084 0,0125 0,0165 0,0167 0,0208 0,0290 0,0641 0,1124 18,0547

Tabla 2 – 12 workers

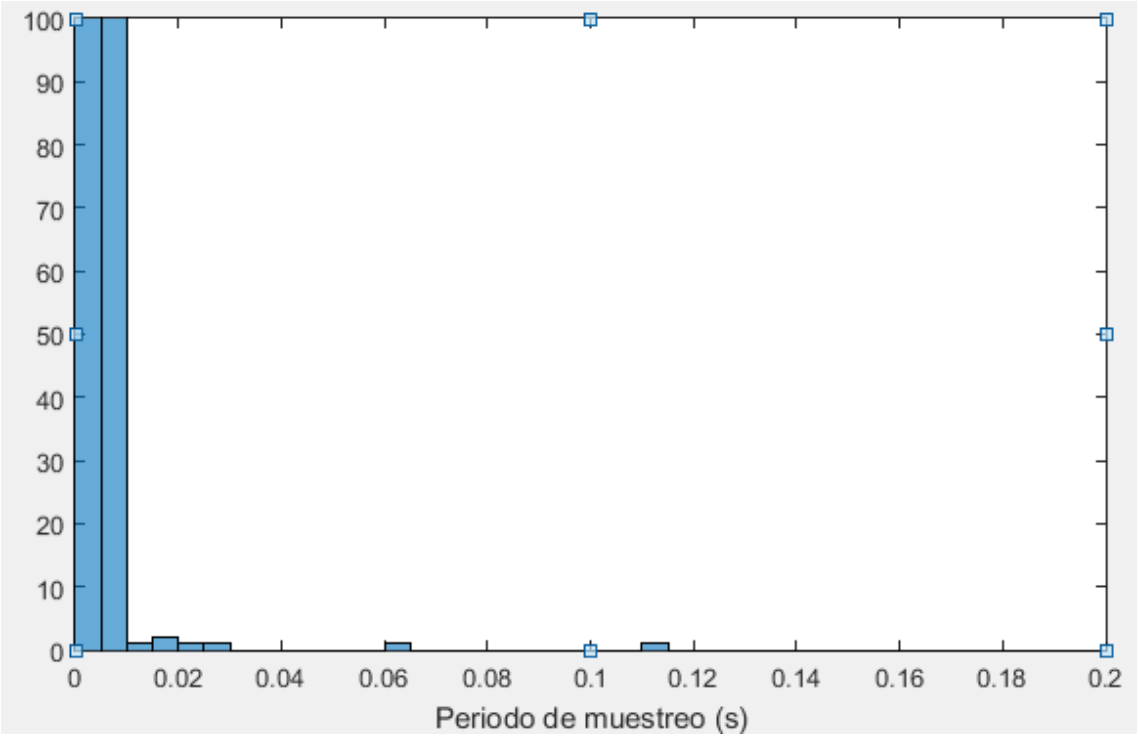

Fig. 18 – Histograma 3 workers + 1 worker/job

c2) Almacenando los datos del sensor cada 5000 muestras recogidas.

- **3 workers + 1 worker/job**

Periodo de muestreo máximo (s)	0,1329
Periodo de muestreo mínimo (s)	0,0004
0.0043 < Nº valores < máximo	867 /4999
Tiempos de muestreo	0,0004 0,0032 0,0040 0,0041 0,0042 0,0082 0,0083 0,0084 0,0125 0,0135 0,0155 0,0156 0,0166 0,0167 0,0176 0,1329

Tabla 3 - 3 workers + 1 worker/job

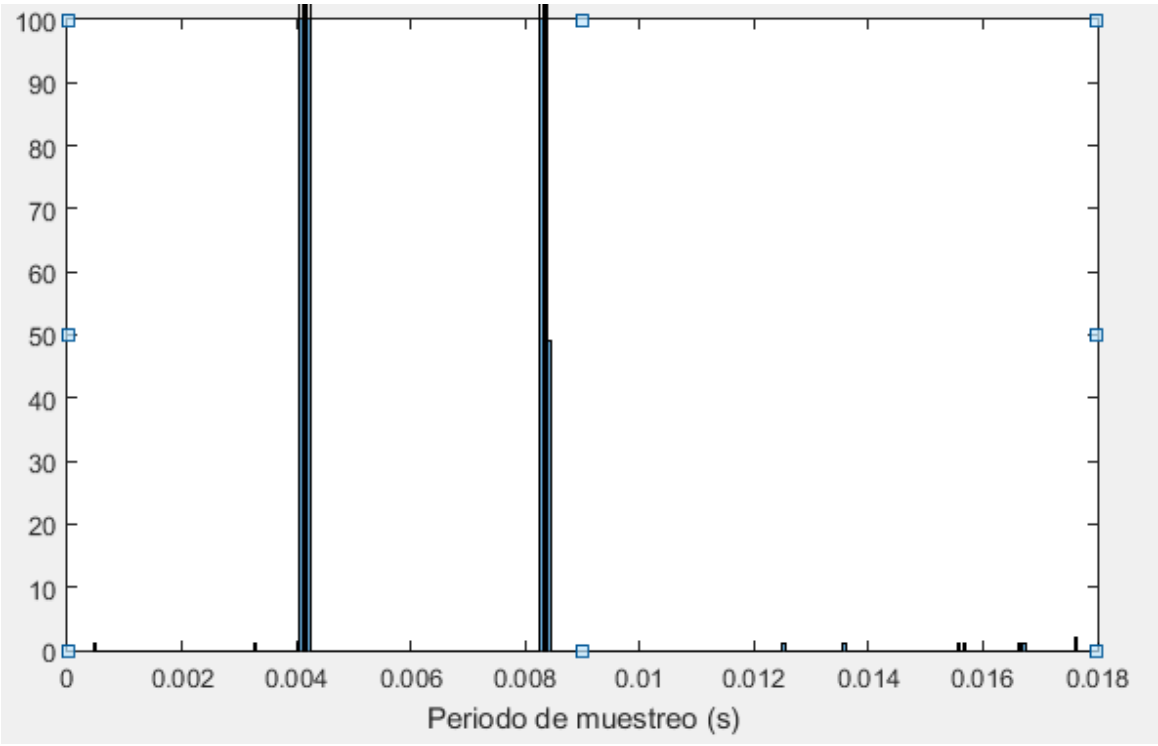

Fig. 19 – Histograma 3 workers + 1 worker/job

- **12 workers**

Periodo de muestreo máximo (s)	0.4799
Periodo de muestreo mínimo (s)	0,0009
0.0043 < Nº valores < máximo	1077/4449
Tiempos de muestreo	0,0009 0,0028 0,0032 0,0040 0,0041 0,0042 0,0082 0,0083 0,0084 0,0125 0,0138 0,0165 0,0166 0,0167 0,0207 0,0240 0,0250 0,0666 0,0708 0,4799

Tabla 4 - 12 workers

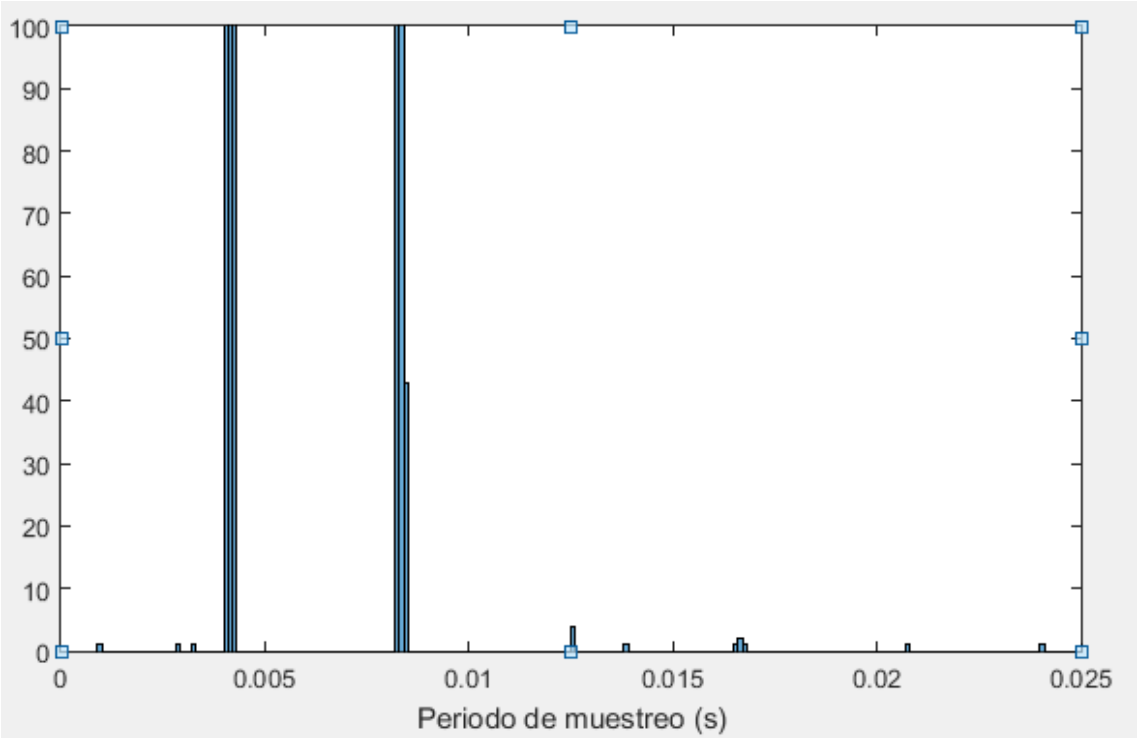

Fig. 20 – Histograma 12 workers

- **12 workers sin guardado de imágenes**

Periodo de muestreo máximo (s)	0.1510
Periodo de muestreo mínimo (s)	0.0031
0.0043 < Nº valores < máximo	219/4449
Tiempos de muestreo	0,0031 0,0040 0,0041 0,0042 0,0082 0,0083 0,0084 0,0124 0,0125 0,0167 0,0207 0,0209 0,0250 0,0290 0,0291 0,0292 0,1509

Tabla 5 - 12 workers sin guardado de imágenes

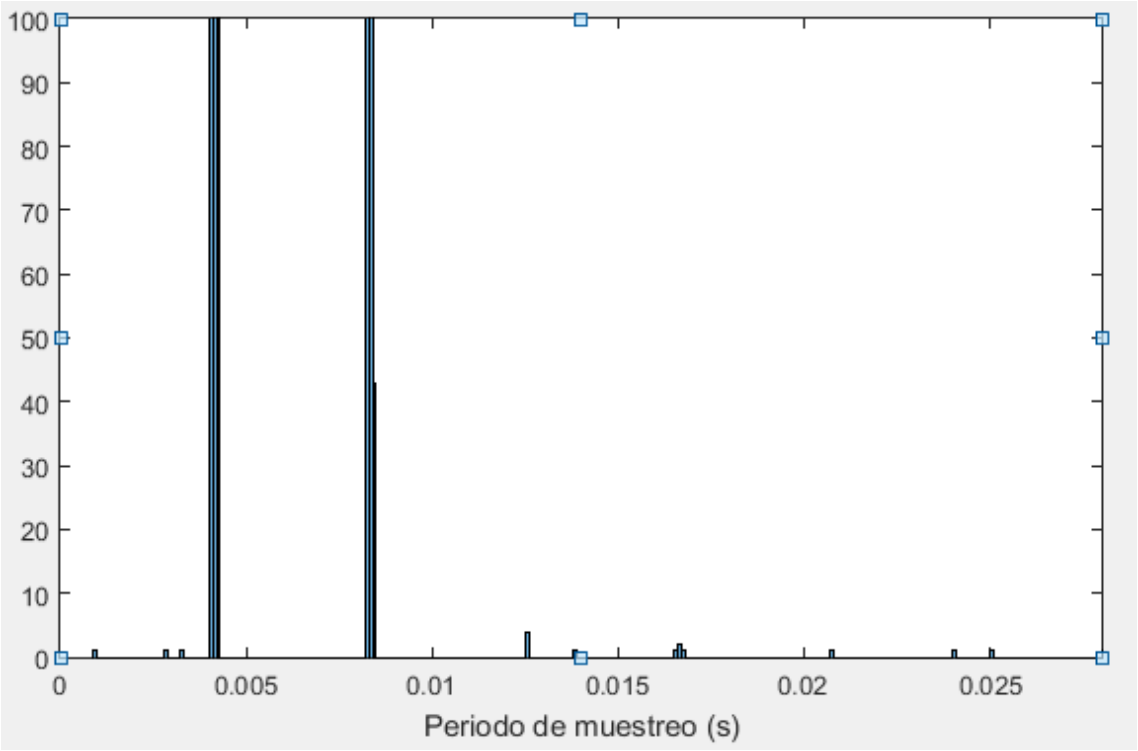

Fig. 21 - 12 workers sin guardado de imágenes

- **12 workers sin comprobar la existencia de la última imagen para terminar la recogida de datos del sensor**

Periodo de muestreo máximo (s)	8,9791
Periodo de muestreo mínimo (s)	0,0006
0.0043 < Nº valores < máximo	1500/4449
Tiempos de muestreo	0,0006 0,0016 0,0030 0,0032 0,0033 0,0036 0,0039 0,0040 0,0041 0,0042 0,0076 0,0082 0,0083 0,0084 0,0094 0,0109 0,0158 0,0165 0,0166 0,0208 0,0243 0,0250 0,0332 0,0375 0,0417 0,0883 0,4795 0,4796 0,4797 0,4800 0,4915 2,4572 3,1361 8,2131 8,4787 8,5629 8,6087 8,9791

Tabla 6 - 12 workers sin comprobación de existencia de la última imagen

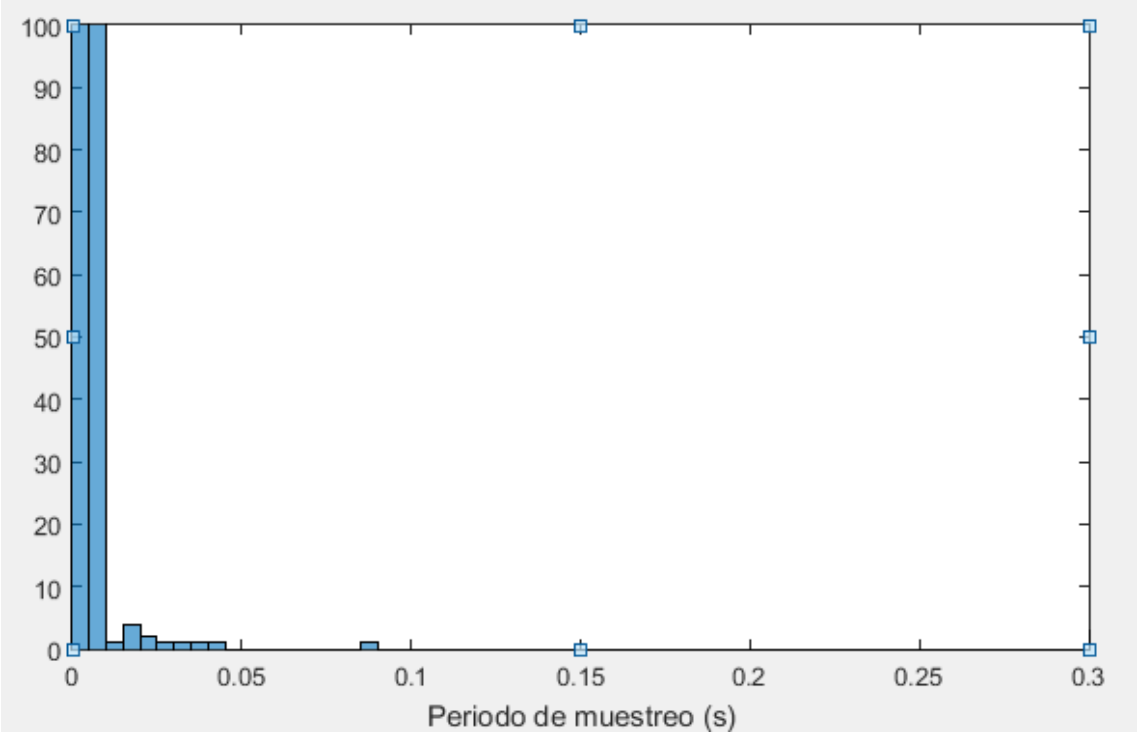

Fig. 22 – Histograma 12 workers sin comprobación de existencia de la última imagen.

- **3 workers sin comprobar la existencia de la última imagen para terminar la recogida de datos del sensor + 1 worker/job**

Periodo de muestreo máximo (s)	0.4914
Periodo de muestreo mínimo (s)	0.0026
0.0043 < Nº valores < máximo	1333/4449
Tiempos de muestreo	0,0026 0,0027 0,0034 0,0040 0,0041 0,0042 0,0082 0,0083 0,0084 0,0090 0,0125 0,0139 0,0167 0,0208 0,0209 0,1723 0,4914

Tabla 7 – 3 workers sin comprobar la existencia de la última imagen + 1 worker/job.

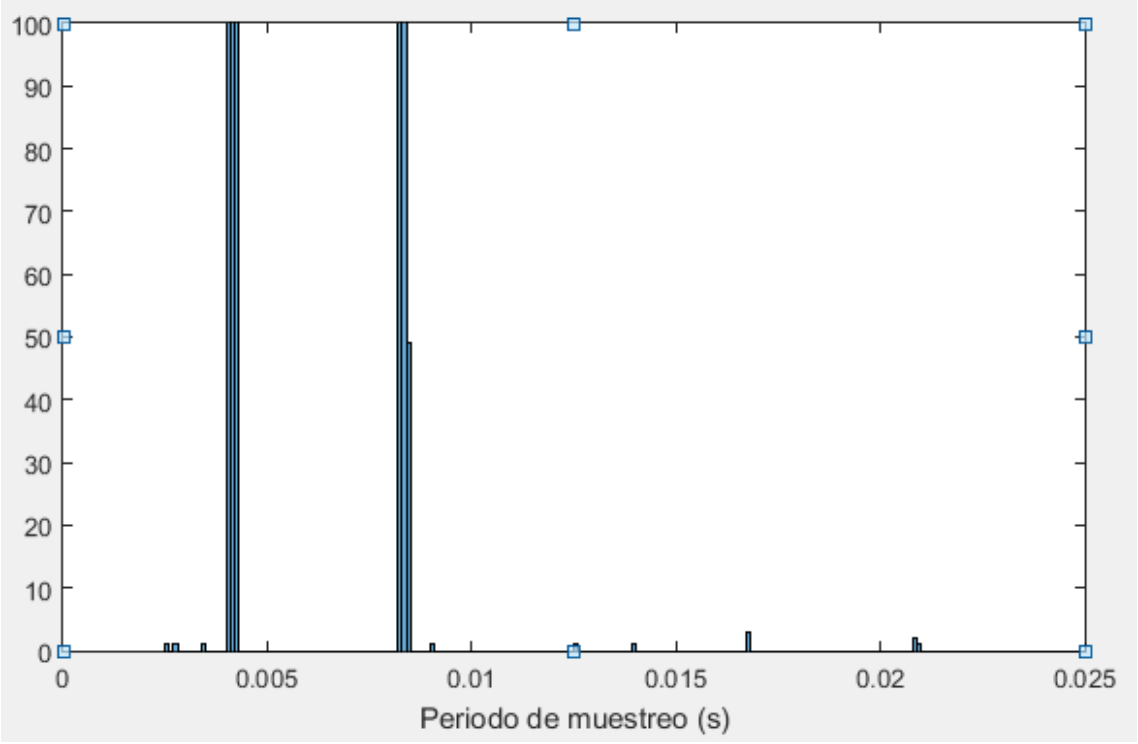

Fig. 23 – Histograma 3 workers sin comprobar la existencia de la última imagen + 1 worker/job

Ante estos resultados aleatorios resulta muy difícil llegar a conclusiones, ya que se aprecian picos altos de muestras tal y como se espera en 0.0042, pero también en 0.0084 lo que supone una pérdida de una muestra, y se llegan a perder en ciertos casos muchas muestras siendo esto inviable para poder sincronizar el sensor con las imágenes correctamente.

Finalmente, dado que el sensor también funciona para Windows 8 se opta por probar el algoritmo en este sistema operativo utilizando además Matlab 2016 para observar cómo afecta este cambio.

El proceso seguido se explica en el siguiente apartado.

4.6 Algoritmo en Windows 8 y Matlab 2016

En esta configuración se comprueba una mayor estabilidad del algoritmo ya que para las adquisiciones de hasta 17 puntos no hay pérdida de muestras prácticamente, y para las de 64 y 65 puntos ésta se acota considerablemente teniendo un patrón claro, tal y como se muestra a continuación en diferentes pruebas realizadas.

4.6.1. Sesión de 65 puntos + 3 workers + 1 worker/job

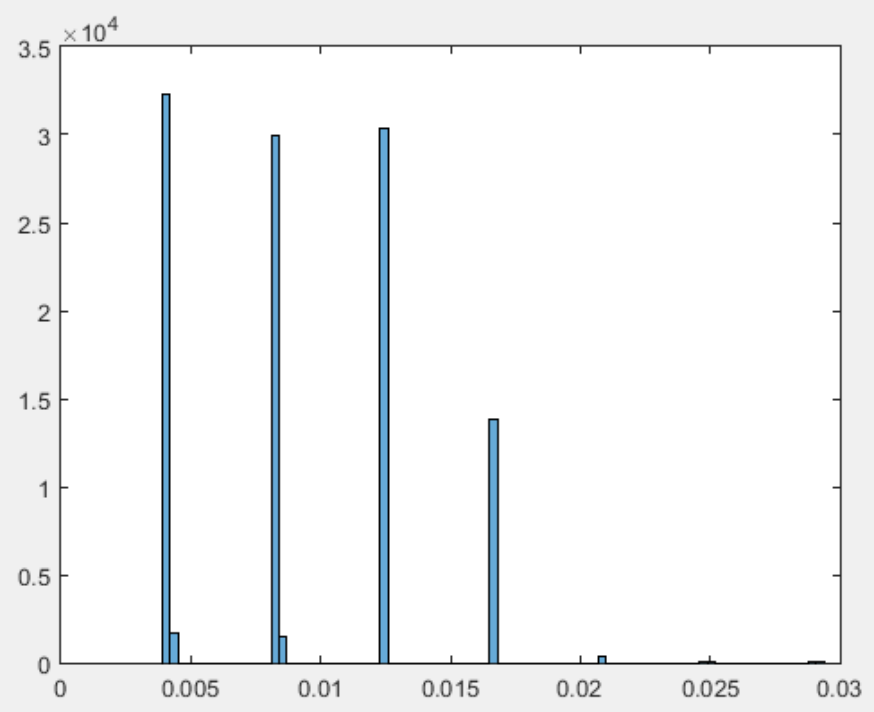

Fig. 24 – Histograma sesión de 65 puntos + 3 workers + 1worker/job

4.6.2. Sesión de 65 puntos + 12 workers + 4 workers/job

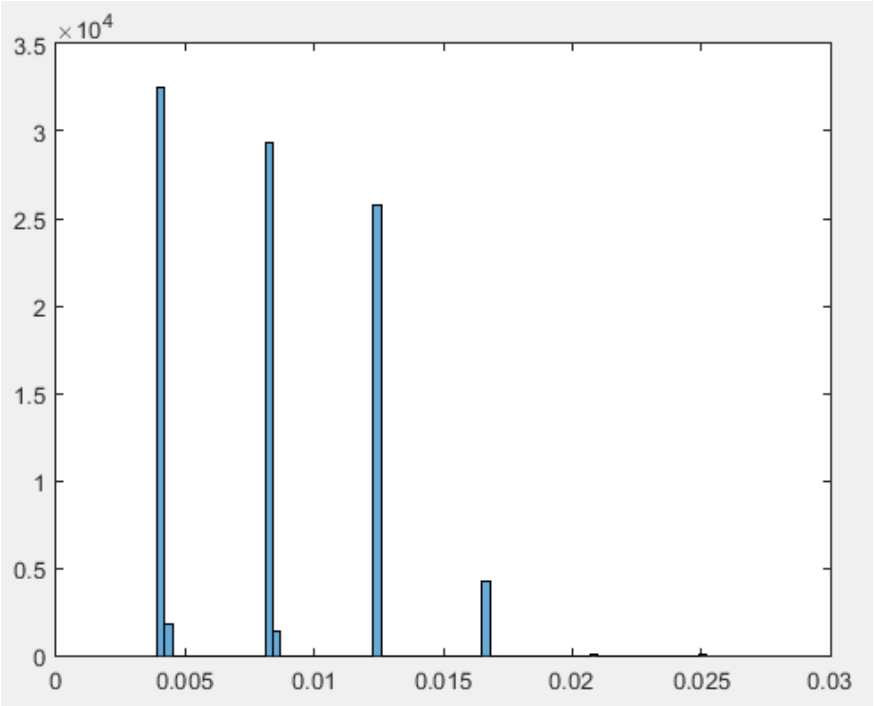

Fig. 25 – Histograma sesión de 65 puntos + 12 workers + 4workers/job

4.6.3. Sesión de 65 puntos + 12 workers + 1 job/worker

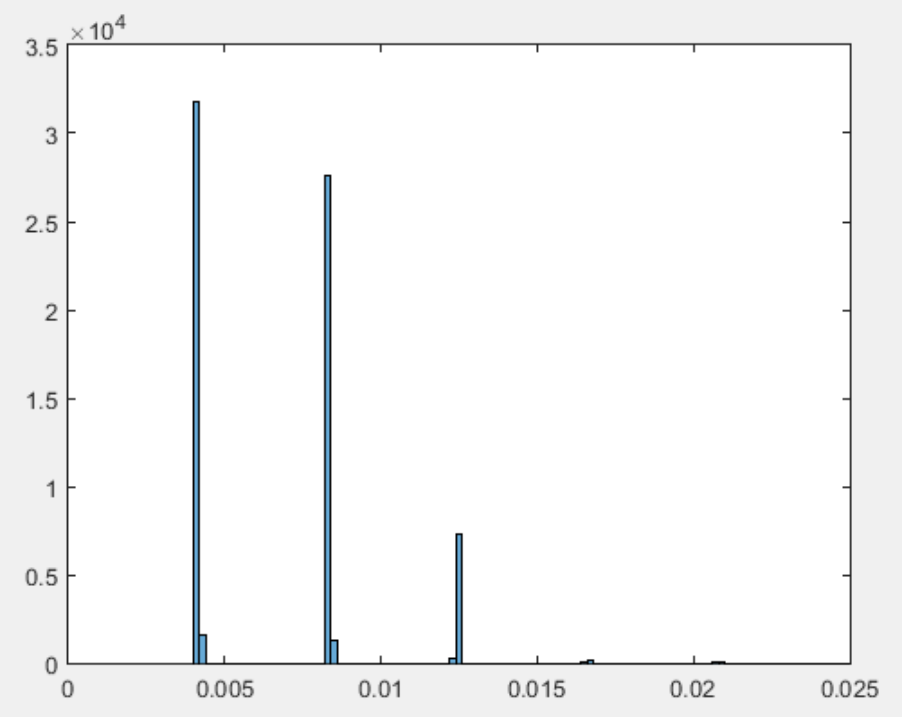

Fig. 26 – Histograma sesión de 65 puntos + 12 workers + 1workers/job

Se observa que la pérdida de muestras se reduce para la configuración de 12 workers + 1 job/worker, por lo tanto, será esta la que se utilice para todas las adquisiciones. Esta muestra un patrón claro de pérdida progresiva: hay un pico alto de muestras recogidas cada 0.0084 segundos, perdiendo 1 muestra con respecto a nuestra frecuencia de muestreo original. En el siguiente pico se pierden 3 muestras y los dos últimos son muy pequeños con respecto al total.

Para estudiar hasta qué punto esta pérdida influye en la sincronización del sensor con las imágenes se construye una matriz n x m siendo n el número de imágenes tomadas por cada punto (30) y m el número de índices a mirar. Se almacenará el valor diferencia de tiempo entre dos muestras consecutivas para cada imagen e índice, pudiendo de esta manera estimar las muestras perdidas.

Para la sesión de 65 puntos, siendo esta la más crítica, observamos que siempre se empiezan a perder muestras a partir del índice 16-17 y conforme aumentan los índices se pierden mayor número de muestras de manera progresiva. Aproximadamente en el punto 56 se producen pérdidas en todas las interpolaciones con las imágenes de ese punto. Esto hace que la sincronización sea muy mala perdiendo información.

A continuación, se muestra la matriz descrita observándose dicha progresión:

		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
7	0	0	0	0	0	0	0	0	0	0.0084	0	0	0	0	0	0.0083	0.0083	0
8	0	0	0	0	0	0	0	0.0083	0.0083	0	0.0083	0	0	0.0083	0	0	0.0084	0.0083
9	0	0	0	0	0	0	0	0	0.0083	0	0	0	0	0.0083	0	0	0.0083	0.0084
10	0	0	0	0	0	0	0	0	0	0.0084	0.0084	0	0	0	0	0	0.0083	0
11	0	0	0	0	0	0	0	0.0084	0.0083	0	0	0.0083	0.0083	0	0	0	0.0083	0.0084
12	0	0	0	0	0	0	0	0	0.0084	0.0084	0.0084	0	0	0.0083	0	0.0084	0.0083	0.0083
13	0	0	0	0	0	0.0083	0	0	0.0083	0.0083	0	0.0083	0.0084	0	0	0.0083	0.0083	0.0084
14	0	0	0.0083	0.0083	0.0083	0	0	0	0	0	0.0083	0	0.0083	0.0083	0.0083	0.0083	0	0.0125
15	0	0	0.0083	0.0083	0	0.0083	0	0	0.0083	0.0083	0	0.0083	0.0083	0.0083	0.0083	0.0083	0	0.0084
16	0	0	0	0.0083	0	0	0.0084	0	0	0	0	0.0083	0.0083	0.0083	0	0.0083	0.0083	0.0083
17	0	0	0	0	0	0	0	0.0084	0	0	0	0.0084	0.0083	0	0.0083	0	0.0084	0.0083
18	0	0	0	0	0.0084	0	0	0	0	0.0083	0.0083	0	0.0084	0.0083	0	0.0084	0.0084	0
19	0	0	0	0.0083	0.0084	0	0.0084	0	0	0	0	0	0.0083	0.0083	0.0084	0	0	0.0084
20	0	0	0	0	0.0083	0	0	0	0	0	0	0	0.0083	0	0.0083	0.0083	0.0083	0.0083
21	0	0	0	0	0	0	0	0.0083	0	0.0083	0.0084	0.0083	0	0	0	0.0083	0	0.0083
22	0	0	0	0	0	0	0	0	0.0084	0.0083	0	0	0	0.0083	0	0.0083	0	0.0084
23	0	0	0	0	0	0	0	0	0	0	0	0.0083	0	0	0	0.0084	0	0.0083
24	0	0	0	0	0	0	0.0083	0	0	0.0083	0	0	0.0083	0	0	0	0	0.0083
25	0	0	0	0	0	0	0	0	0	0.0083	0.0083	0.0084	0	0	0	0.0084	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0084	0	0.0083	0	0
27	0	0	0	0	0	0	0	0.0083	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0.0083	0.0083	0	0	0	0	0.0083	0	0.0083
29	0	0	0	0	0	0.0083	0	0	0.0084	0	0.0083	0	0.0083	0	0.0083	0.0083	0	0.0084
30	0	0	0	0	0	0	0	0.0083	0.0083	0.0084	0	0.0084	0	0.0083	0.0083	0.0083	0	0.0083

Fig. 27 - Muestra de la matriz para cada imagen capturada en la sesión de 65 puntos

4.6.2. Sesión 65 puntos aplicación original

Llegados a este punto, se ejecuta la aplicación original "***runtracker.m"*** para la duración de lo que sería una adquisición de 65 puntos (4 min aprox.) y se comprueba que esta desviación de la frecuencia de muestreo es inherente a la aplicación original empezando la pérdida de muestras a partir del minuto.

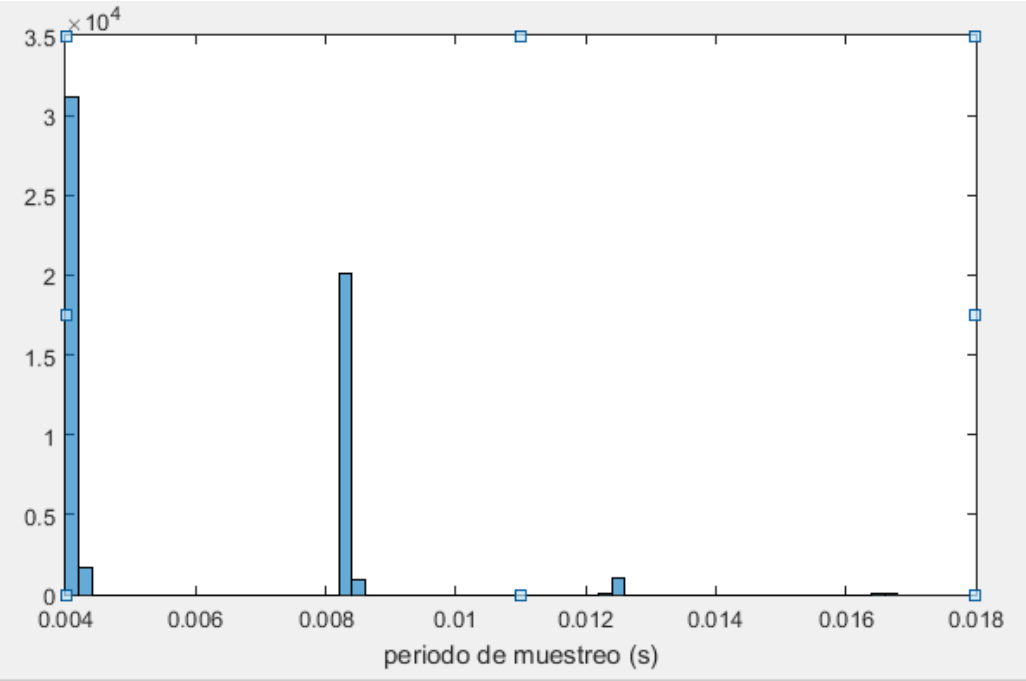

Fig. 28 – Patrón de pérdida de muestras propio de la aplicación original del TrakStar

Por lo tanto, ante estos resultados, finalmente se propone hacer la adquisición de 64 y 65 puntos en franjas de índices menores a 16 índices para mantener el sincronismo. De esta manera la sesión de 64 se divide en 8 franjas de 8 puntos y la de 65 en 5 franjas de 13 puntos cada una.

Para poder implementar esta solución resulta necesario que el Trakstar pare de recoger muestras al final de cada franja de puntos una vez almacenada la información de posición de estos y acto seguido empiece una nueva adquisición, sin necesidad de reiniciarse, ya que hacer el setup supondría un tiempo de unos 15 segundos hasta que comience de nuevo a adquirir. Para ello se ejecuta la aplicación original del transmisor para comprobar si realmente es imprescindible entre adquisiciones hacer dicho reinicio. Se hace una adquisición y seguidamente otra, comprobando de nuevo si la frecuencia de muestreo se mantiene constante, en la información de la última se observa que al inicio en la muestra numero 7 siempre hay una pérdida considerable, pero al ser siempre en la misma posición y al inicio no supone un gran problema.

Para mantener el sincronismo es necesario que al terminar cada franja de puntos se compruebe la existencia del correspondiente archivo con los datos del sensor almacenado en la carpeta del usuario como ***"data_sen.mat"*** y una vez comprobado

esperar un tiempo de 6 segundos para que al sensor le dé tiempo a grabar a disco las muestras recogidas y comenzar la adquisición de datos del siguiente grupo de puntos en sincronía a la captura de imágenes.

Se implementa una nueva función paralela para esta recogida de muestras del Trakstar de las sesiones de 64 y 65 puntos llamada ***"get_points_trak_star_64p.m"*** de manera que se comprueba la existencia de la última imagen de la franja correspondiente cada 960 muestras recogidas para con ello terminar la adquisición, almacenar el correspondiente .mat y una vez comprobado que no existe aún la matriz de imágenes correspondiente al último punto de la sesión empezar a recoger muestras del siguiente grupo de puntos. Y así para el resto de franjas hasta finalizar la sesión.

Resulta imprescindible hacer la comprobación cada 960 muestras para que se mantenga la sincronía, ya que cualquier otro valor hace que el sensor se desestabilice perdiendo muestras y con ello el sincronismo con la captura de imágenes.

Resulta curioso que de esta manera la pérdida que se producía debido a no reiniciar el transmisor entre capturas en la aplicación original del transmisor, con esta configuración en Matlab no ocurre consiguiendo mantener la frecuencia de muestreo original estable durante toda la sesión y con ello el sincronismo sensor-imágenes.

4.6.3. Problema optimización en la calibración cámara-sensor

Se lleva a cabo una nueva calibración siguiendo su correspondiente proceso descrito al inicio de esta memoria (**apartado 3.3.1**) en Windows 8 y Matlab2016 para comprobar que funciona correctamente el programa. A continuación, se introduce un nuevo usuario y posteriormente al marcado de los corners y la punta de la nariz en el preview online observamos cómo estos puntos presentan una desviación hacia la derecha del usuario y hacia arriba. Para asegurar que la relación entre el sensor y la punta del útil es correcta se ejecuta la función de optimización ***"calcular_punta_marcador_optimizacion"***.

A pesar de esta optimización, el error persiste, siendo similar al que se produce previamente a la optimización de la calibración cámara-sensor tal y como se desarrolla en [3].

Para asegurarnos de que el marcado es correcto se hace sobre el damero, siendo más preciso que el marcado de los corners y la punta de la nariz.

Para ver el efecto de la optimización se compara el preview online entre el marcado sin optimizar y con la optimización una vez calibrado el sistema correctamente.

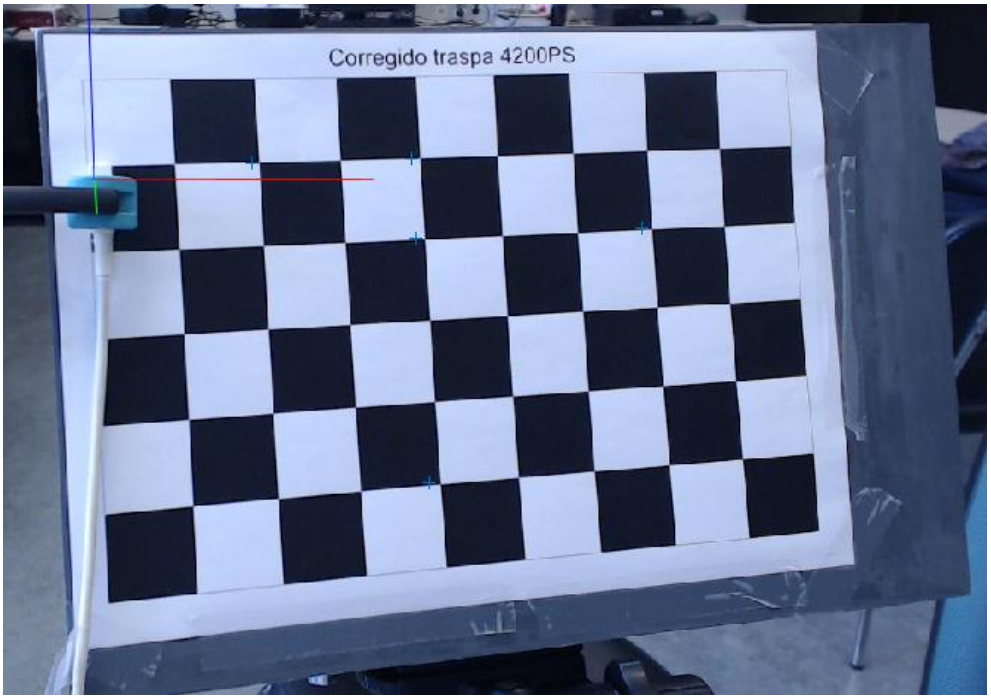

Fig. 29 – Preview online sin optimización

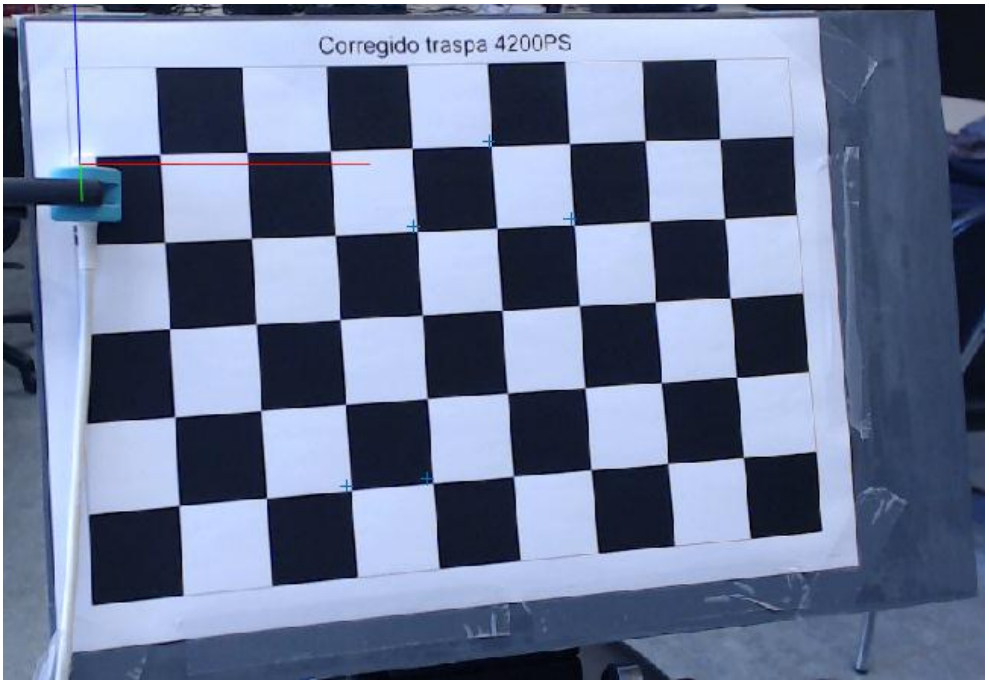

Fig. 30 – Preview online con optimización

Se observa que la optimización corrige el desvío hacia la izquierda de los puntos, pero éstos siguen estando en cierta medida a la derecha y hacia arriba, por tanto, la optimización no es del todo correcta. El centro de los ejes representados también presenta la misma desviación de los puntos, por lo que el desajuste reside en la calibración del sensor1-cámara o en la optimización, ya que el marcado del sensor 2 depende de la calibración del 1.

Se vuelve a calibrar para asegurarnos de que la calibración de la cámara es correcta y no introduce error en el posterior marcado. Por tanto, el objetivo es obtener un error de calibración pequeño y mover el damero con suficiente variación tanto de pitch como de yaw (siempre teniendo en cuenta la limitación en cuanto a distancia del sensor con el transmisor). También se presta especial atención a que el sensor colocado en el damero sujetado por la pinza se permanezca en la misma posición durante todo el proceso.

Se obtiene lo siguiente,

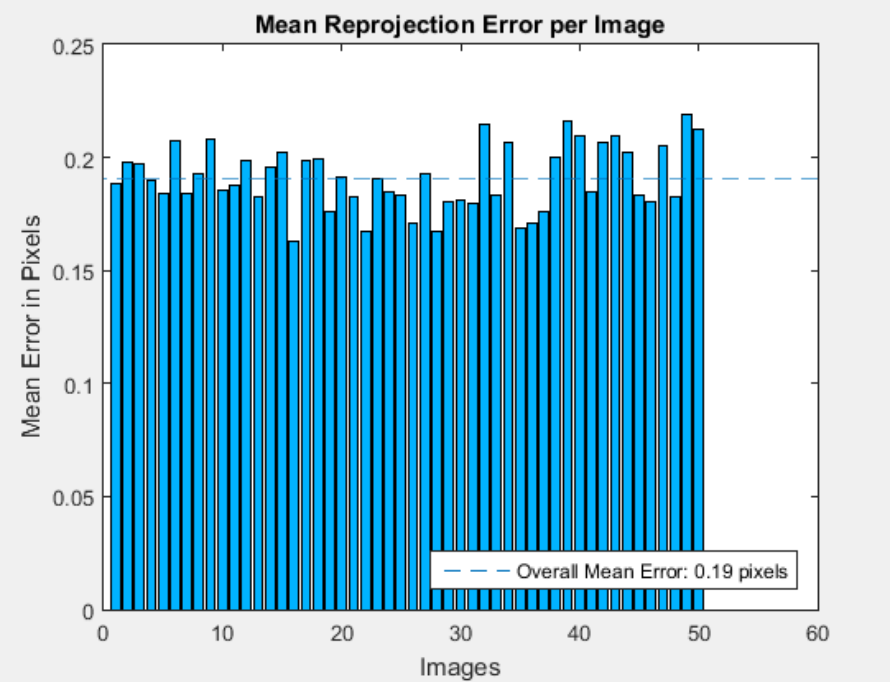

Fig. 31 – Error obtenido en el proceso de calibración de la cámara

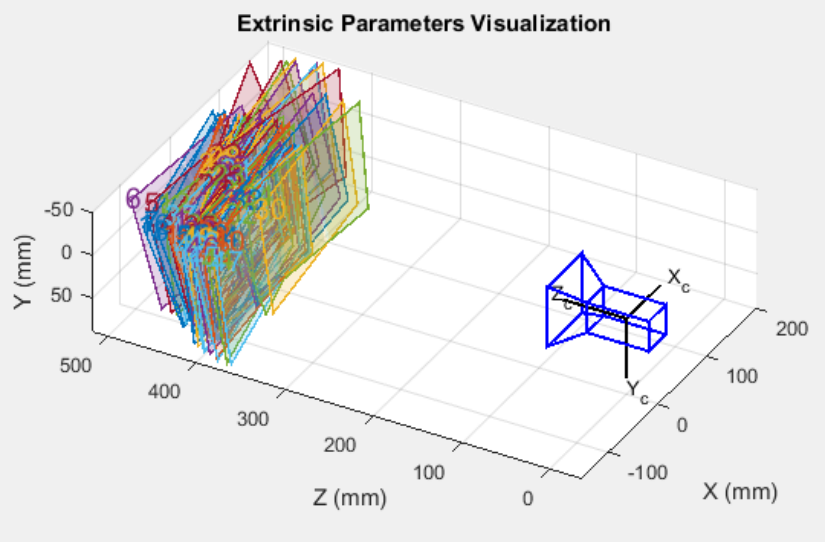

Fig. 32 – Visualización parámetros extrínsecos

Se comprueba durante el proceso de optimización de la calibración que se proyecta correctamente tanto el centro del sensor 1 como los puntos interiores del damero, tal y como se muestra en la siguiente imagen,

Fig. 33 – Proyección optimización

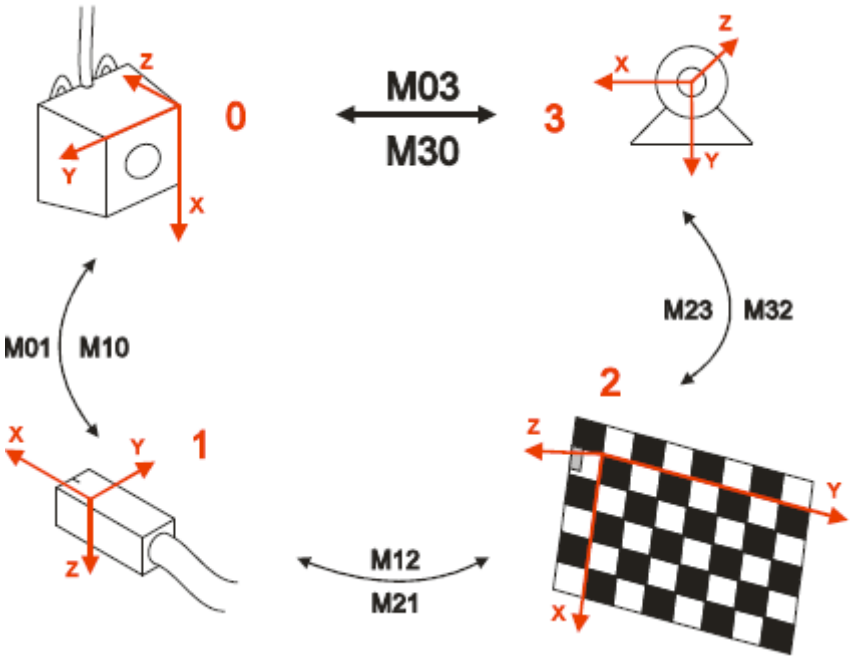

Fig. 34 – Elementos del proceso de calibración [3]

Teniendo en cuenta la relación entre los sistemas de referencia, en la función correspondiente al preview online se representan los ejes del sensor 1 referenciados al sistema de coordenadas de la cámara haciendo la transformación de los sistemas de referencia sensor-transmisor-cámara. Para corroborar que el centro del s1 con respecto a la cámara es correcto lo representamos con un asterisco mediante la transformación M12-M23.

Tal y como se muestra el origen del sensor representado por ambos caminos coincide, y se continua obteniendo el mismo error de marcado comentado inicialmente,

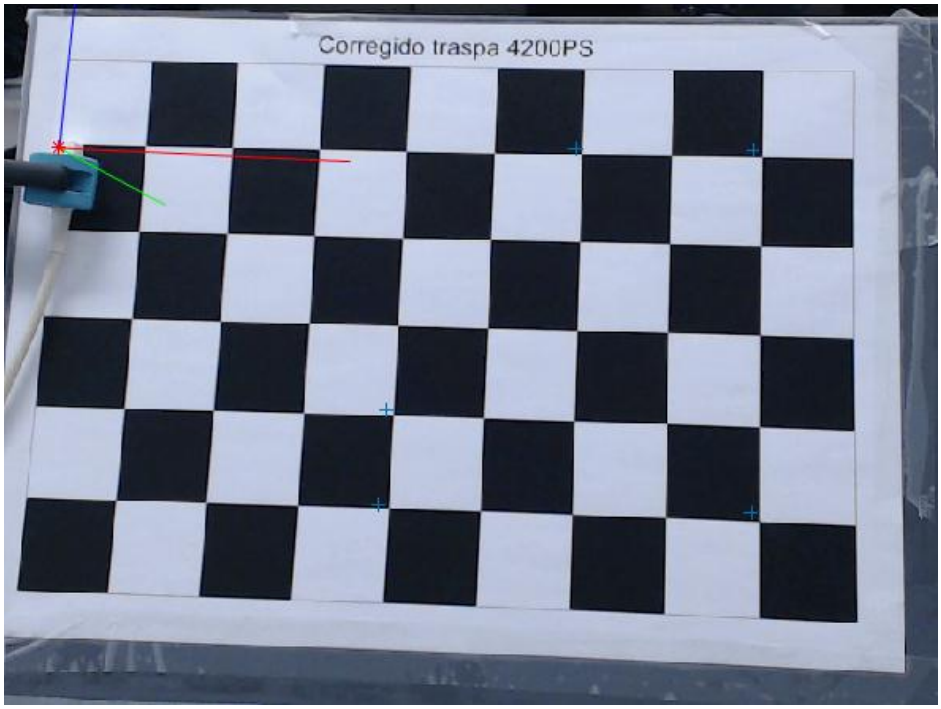

Fig. 35 – Preview online proyectando el centro de S1

A continuación, se calibra de nuevo, esta vez girando ligeramente hacia abajo la webcam se obtiene un error de 0.19 pixeles, como en el caso anterior. A continuación, se muestra el marcado

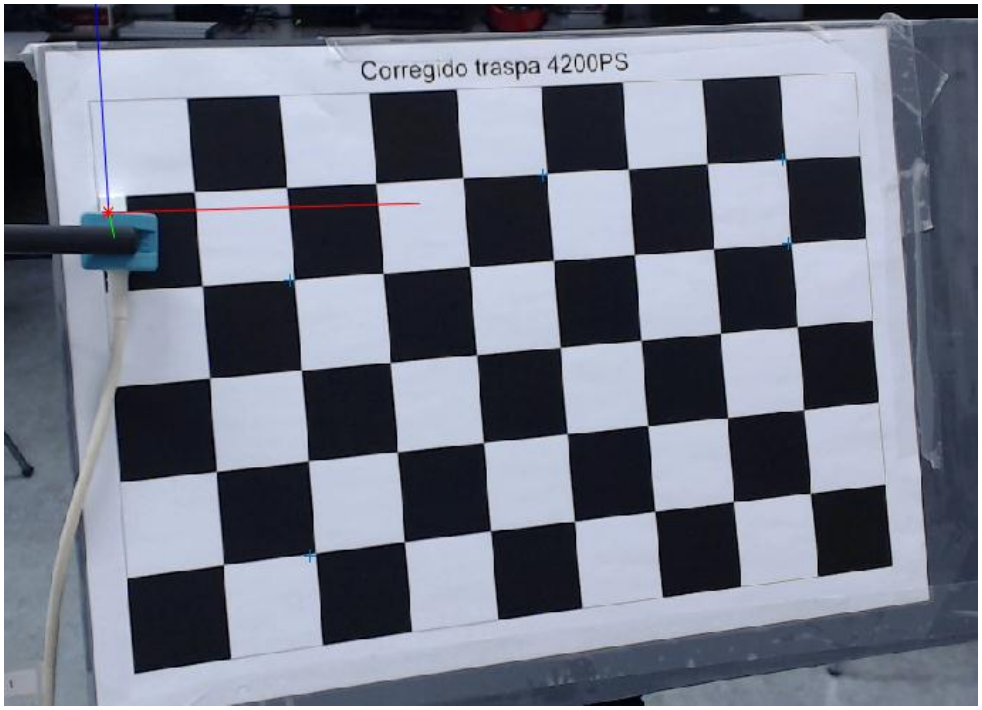

Fig. 36 – Preview online marcado damero

En este caso se aprecia cómo aparece únicamente la desviación hacia la izquierda desapareciendo la desviación hacia arriba.

Se vuelve a calibrar bajando algo más la cámara, obteniendo de nuevo un error bajo (0.2 pixeles)

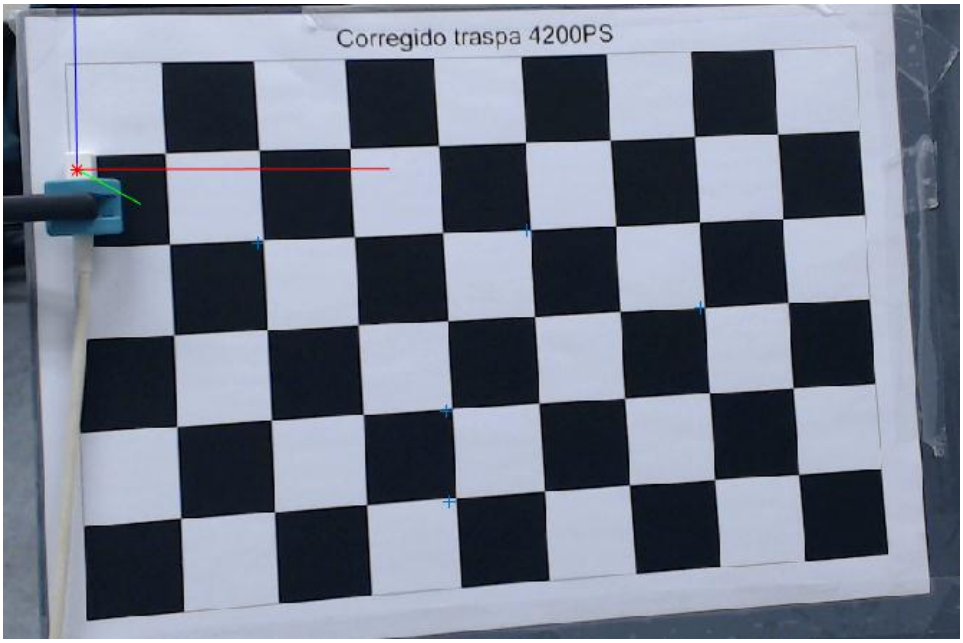

Fig. 37 – Preview online marcado damero

Se observa que el patrón persiste, de manera que finalmente se propone corregirlo añadiendo un offset después de la optimización de 2 mm en el eje Y del damero, obteniendo el resultado deseado, tal y como se muestra a continuación,

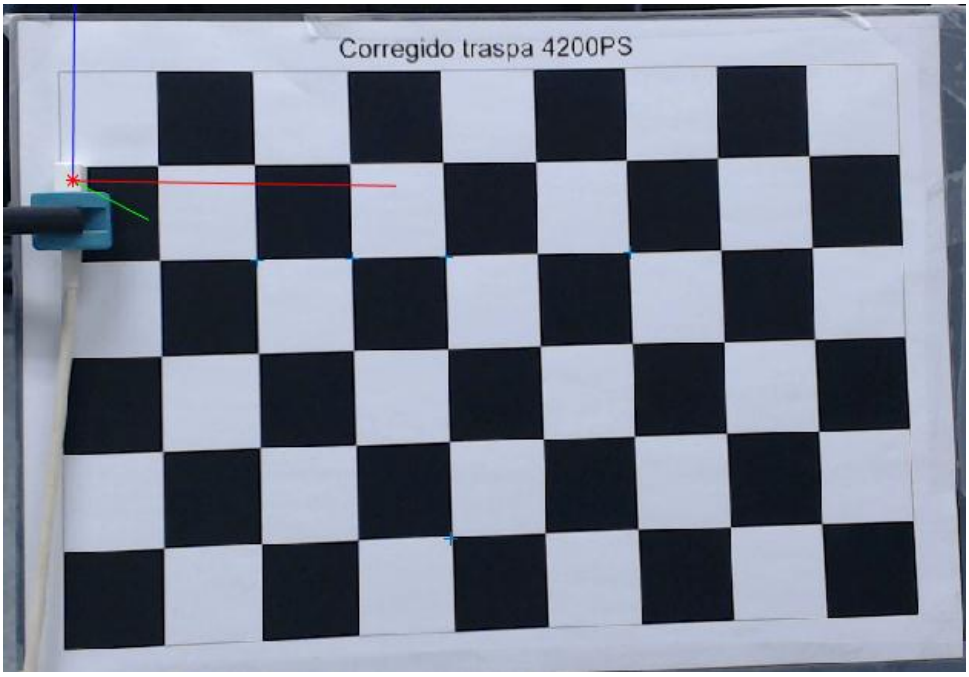

Fig. 38 – Preview online marcado damero + offset 2 mm.

A continuación se comprueba el marcado sobre la cabeza de prueba,

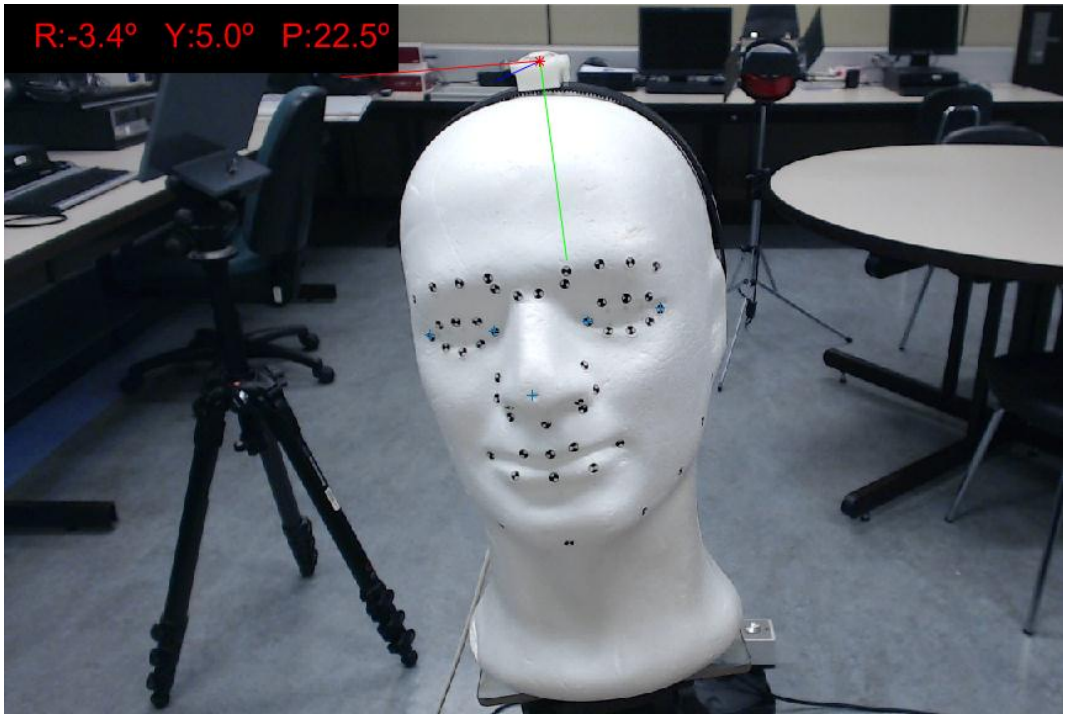

Fig. 39 – Preview online marcado cabeza + offset 2 mm.

5. DESARROLLO DE LA BBDD

Debido a los problemas surgidos durante la realización del TFG se hace una BBDD de prueba con 2 usuarios.

5.1. Datos obtenidos en la calibración del sistema

A continuación, se muestra el error obtenido para la calibración, así como la visualización de los parámetros extrínsecos

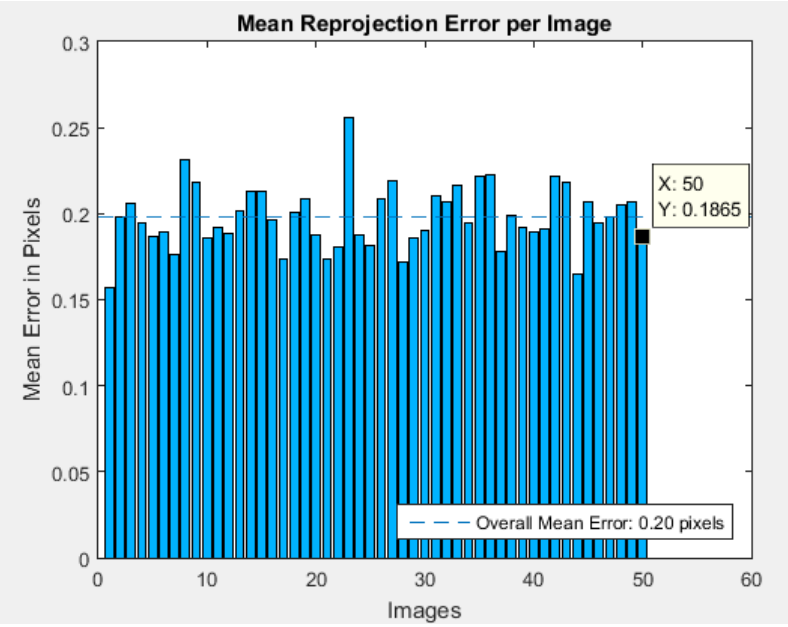

Fig. 40 – Error calibración

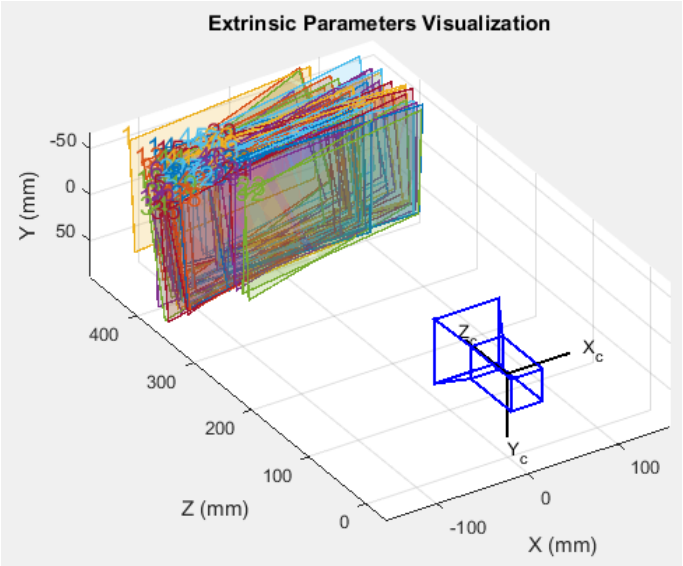

Fig. 41 – Visualización parámetros extrínsecos

5.2. Descripción BBDD

Para el desarrollo de la base de datos se hace uso de los focos anteriormente mencionados y se ajustan manualmente los ajustes de la cámara: exposición, ganancia y autoenfoque de manera que consigamos una iluminación adecuada para la grabación.

Las sesiones planteadas para cada usuario son las siguientes:

- **Sesión 1** ➔ 64 puntos verdes + 1 punto central rojo moviendo la cabeza libremente.

- **Sesión 2** ➔ 16 puntos amarillos + 1 punto central rojo estando el usuario situado a la izquierda a la altura de la segunda columna de puntos y moviendo la cabeza libremente.

- **Sesión 3** ➔ 16 puntos amarillos + 1 punto central rojo estando el usuario situado a la derecha a la altura de la séptima columna de puntos y moviendo la cabeza libremente.

- **Sesión 4** ➔ 16 puntos amarillos + 1 punto central rojo estando el usuario centrado, separado un palmo de la mesa y moviendo la cabeza libremente.

- **Sesión 5** ➔ 16 puntos amarillos + 1 punto central rojo estando el usuario centrado, pegado con la silla a la mesa y moviendo la cabeza libremente.

- **Sesión 6** ➔ 16 puntos amarillos + 1 punto central rojo estando el usuario centrado, pegado con la silla a la mesa y con la cabeza estática.

- **Sesión 7** ➔ 64 puntos verdes + 1 punto central rojo con la cabeza estática.

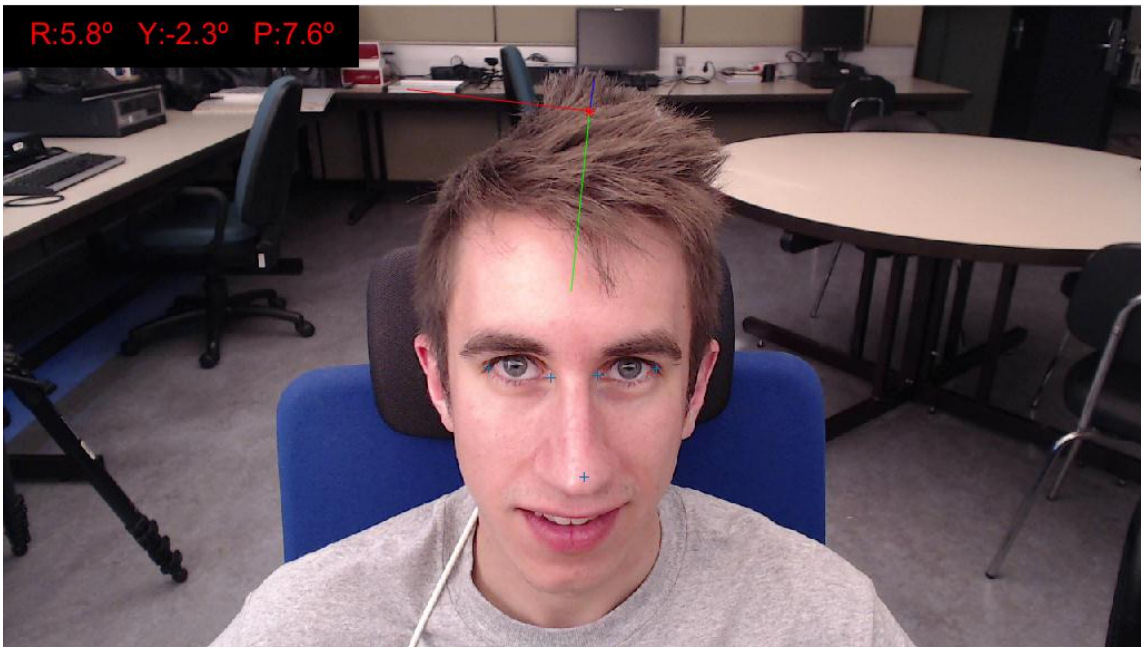

Fig. 42 – Marcado usuario 1

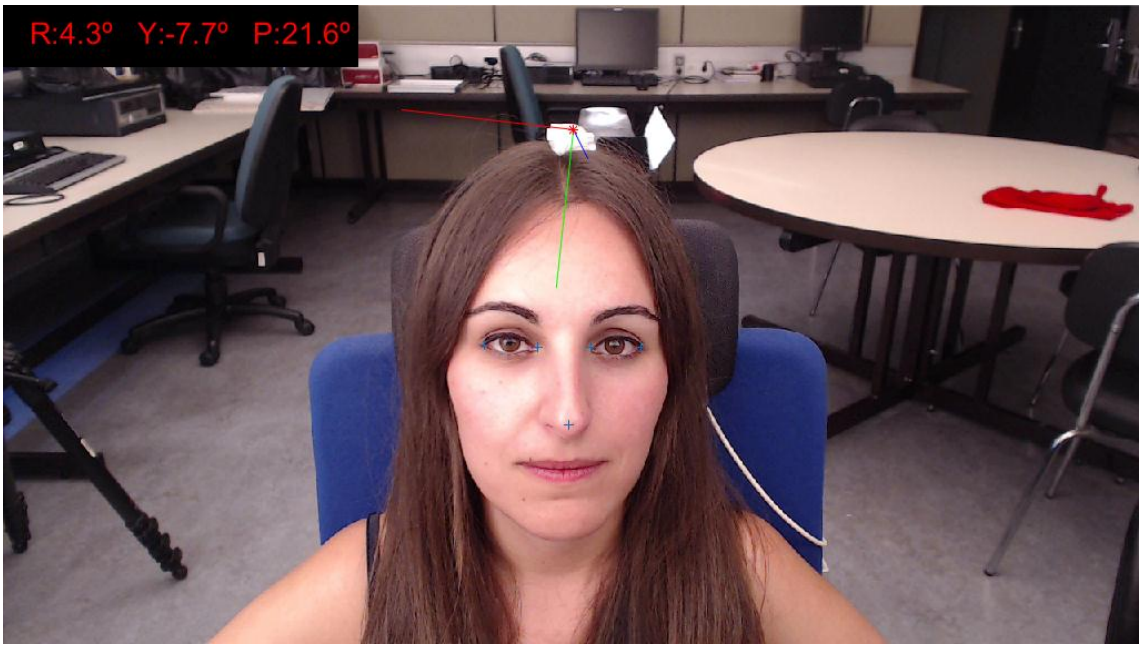

Fig. 43 – Marcado usuario 2

Fig. 44 – Usuario 1 sesión 1

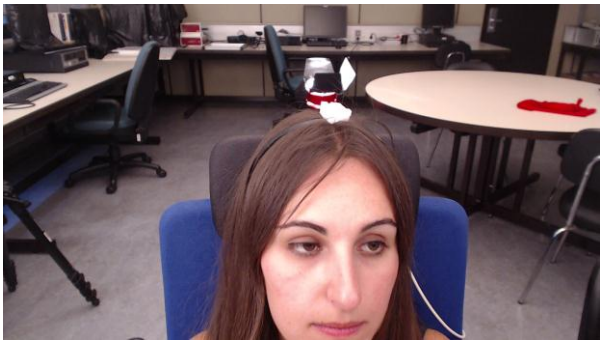

Fig. 45 – Usuario 2 sesión 1

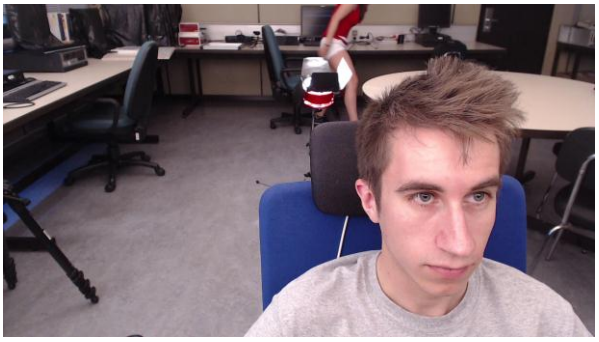

Fig. 46 – Usuario 1 sesión 2

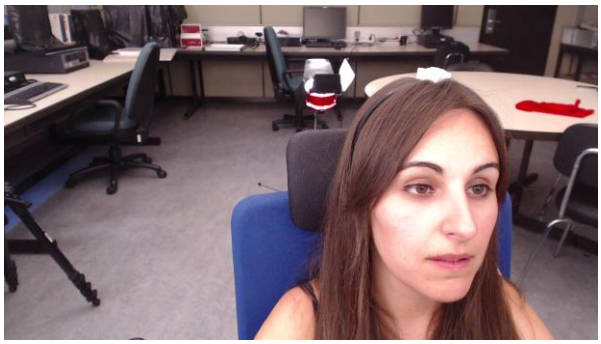

Fig. 47 – Usuario 2 sesión 2

Fig. 48 – Usuario 1 sesión 3

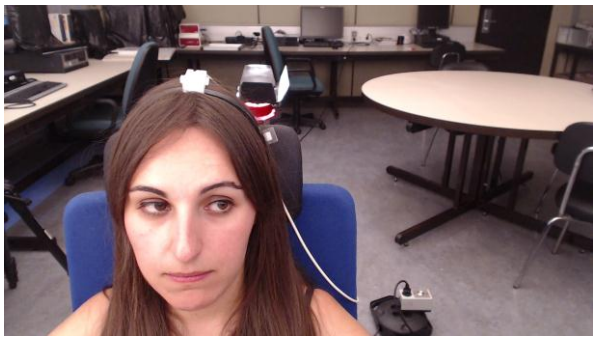

Fig. 49 – Usuario 2 sesión 3

Fig. 50 – Usuario 1 sesión 4

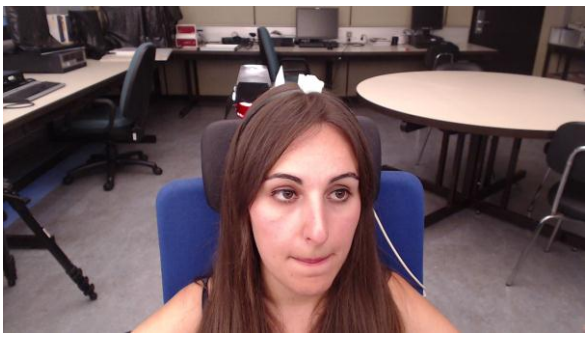

Fig. 51 – Usuario 2 sesión 4

Fig. 52 – Usuario 1 sesión 5

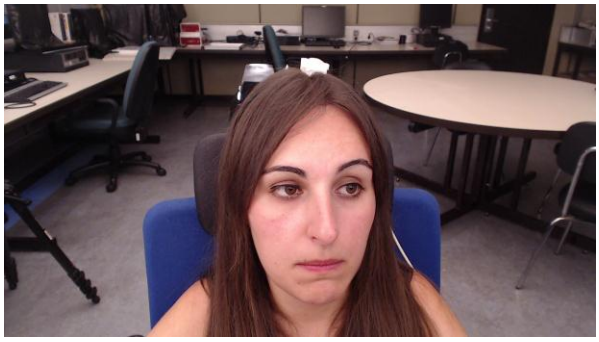

Fig. 53 – Usuario 2 sesión 5

Fig. 54 – Usuario 1 sesión 6

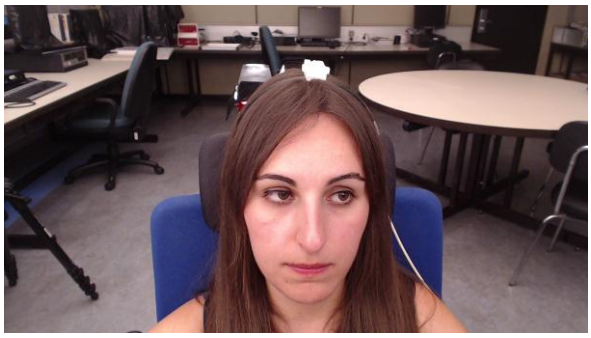

Fig. 55 – Usuario 2 sesión 6

Fig. 56 – Usuario 1 sesión 7

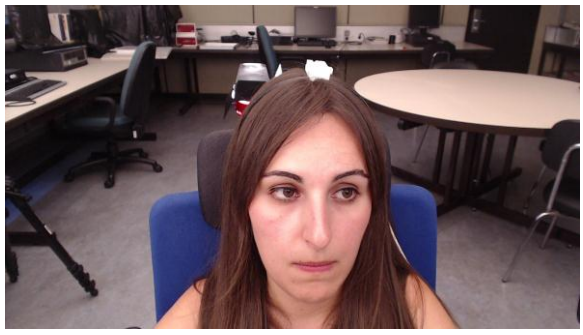

Fig. 57 – Usuario 2 sesión 7

6. CONCLUSIONES

El objetivo de este proyecto es modificar el algoritmo existente para de esta manera implementar una BBDD basada en HPE y POS.

En primer lugar, para poder hacer una calibración buena resulta imprescindible colocar el sensor bien sujeto con la pinza sobre el damero y hacer movimientos de pitch y yaw variados siempre a una distancia razonable del transmisor para mantener una calidad buena.

Siguiendo con la calibración, habría que estudiar mejor el problema en cuanto al marcado ya que para las pruebas realizadas con la cámara inclinada el offset corrige la desviación de los puntos presente en el marcado, pero no es del todo fiable esta solución. Probablemente sea debido a Matlab 2016 ya que en la configuración anterior esto no ocurría.

En segundo lugar, comentar la gran sensibilidad a fallos del sistema inicial a la hora de sincronizar los datos obtenidos por el sensor con la captura de imágenes haciéndolo mediante una doble adquisición (ventana Matlab y terminal en C++). El hecho de unificar este aspecto haciéndolo paralelamente hace que la grabación sea mucho más ligera para el usuario, mucho más rápida (17 segundos/índice vs 1 segundo/índice) y también conseguir eliminar el principal problema de hacer una mala sincronización. De manera que se ejecutan 3 funciones paralelas: la correspondiente al guardado de imágenes, a la posterior conversión de estas a png y a la recogida de datos del sensor.

Por otro lado, también se observa la gran sensibilidad en cuanto a variación de la frecuencia de muestreo del Trakstar. A la vista de los resultados obtenidos, resulta totalmente inviable usar el algoritmo en Matlab2013 y Windows XP. Ha sido necesario mucho tiempo probando diferentes configuraciones para encontrar una configuración estable del sistema.

Esta configuración se ha logrado implementando el algoritmo en Windows 8 y Matlab 2016, paralelizando con 12 workers configurando 1 worker por job en las preferencias de la paralelización y comprobando la existencia de las imágenes cada 290 muestras.

BIBLIOGRAFÍA

[1] E. Murphy-Chutorian y M. Manubhai Trivedi, «Head Pose Estimation in Computer Vision: A Survey,» *IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume:31 , Issue: 4).*

[2] P. Jimenez Molina, «Face pose estimation with automatic 3D model creation for a driver inattention monitoring application».

[3] J. J. Bengoechea Irañeta, «Comparativa de algoritmos de visión monocular para la estimación de la posición de la cabeza.,» 2014.

[4] J. Inda y D. Castillo, «Validación de algoritmos de estimación de la mirada en entornos simulados».